Not Yet Another Digital ID: Privacy-Preserving Humanitarian Aid Distribution

Boya Wang*, Wouter Lueks†, Justinas Sukaitis‡, Vincent Graf Narbel‡, Carmela Troncoso*
*SPRING Lab, EPFL, Lausanne, Switzerland
{boya.wang,carmela.troncoso}@epfl.ch
†CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
lueks@cispa.de
‡International Committee of the Red Cross, Geneva, Switzerland
{jsukaitis,vgraf}@icrc.org

***Abstract*—Humanitarian aid-distribution programs help bring physical goods to people in need. Traditional paper-based solutions to support aid distribution do not scale to large populations and are hard to secure. Existing digital solutions solve these issues, at the cost of collecting large amount of personal information. This lack of privacy can endanger recipients' safety and harm their dignity. In collaboration with the International Committee of the Red Cross, we build a safe digital aid-distribution system. We first systematize the requirements such a system should satisfy. We then propose a decentralized solution based on the use of tokens that fulfills the needs of humanitarian organizations. It provides scalability and strong accountability, and, by design, guarantees the recipients' privacy. We provide two instantiations of our design, on a smart card and on a smartphone. We formally prove the security and privacy properties of these solutions, and empirically show that they can operate at scale.**

***Index Terms*—privacy-preserving technologies, privacy engineering, humanitarian aid distribution**

1. Introduction

Humanitarian organizations, such as the International Committee of the Red Cross (ICRC) [30], aim to protect and assist the victims of violence, famines, and disaster. One of their main operations is the distribution of physical goods, such as food or blankets, in emergency scenarios [27].

Traditionally, humanitarian organizations use paper-based systems to support aid-distribution, e.g., a list with recipients' information and allocation of goods, or paper vouchers valid for particular aid items. These approaches, while practical, have important shortcomings: searching for information on a paper list does not scale beyond a few hundred recipients, vouchers are easily faked, etc.

To address these shortcomings, humanitarian organizations are looking into easy-to-scale digital solutions to support their aid-distribution programs. They are also aware that digitalization should be handled with care, as it brings new risks to the vulnerable populations they serve [36].

Building a digital aid-distribution system that preserves the safety, rights, and dignity of humanitarian aid recipients requires a deep understanding of the humanitarian context. We partner with the ICRC to learn the requirements and constraints associated with distributing aid in emergencies. Our interactions reveal the following challenges:

1) *Secure household-oriented aid.* Aid-distribution systems must permit aid allocation per household (i.e., a domestic unit of several members sharing meals and income), yet they must ensure that households can only request aid once per distribution round (e.g., per month).
2) *Avoid reliance on powerful hardware and connectivity*. Most aid-distribution programs take place in crisis-affected settings where we cannot assume the existence of last-generation hardware or internet connectivity.
3) *Auditability*. For accountability reasons, humanitarian organizations need to prove that aid is distributed in an honest manner, i.e., only to legitimate recipients.
4) *Strong privacy.* Aid-distribution systems must avoid causing digital harm to the individuals [10]. The system must avoid generating databases with recipients' data and creating digital traces related to recipients' actions.

Existing digital aid-distribution solutions can address the first three challenges. Often, they achieve this by integrating an Identity Management System into their solution [58]. This creates (central) databases with the personal data of recipients – and even more sensitive information if the program requires strong authentication, such as the UN Refugee Agency's biometric identification system for refugees [53], or Pakistan's biometric-based Watan Card [43].

These solutions' reliance on data, however, conflicts with the strong need for privacy (challenge 4) and makes them ill-suited for the highly-sensitive humanitarian context [25]. Not only can they jeopardize the safety of recipients [20], [26], but they may also complicate the relationship of humanitarian organizations with local authorities, which shades the neutrality of the humanitarian actors [50]. For example, in Yemen, the World Food Program clashed with Houthi authorities because of the disagreement over the usage and control of biometric data [14]. Finally, from an ethical perspective, it is questionable whether gathering personal information of vulnerable people is acceptable given the risks that it entails for them [23], [50].

The reliance of data in existing systems is inherent to their approach to prevent distribution to illegitimate recipients and to ensure accountability, mainly based on centralizing the collection of logs. Privacy risks in such centralized solutions can only be avoided by using expensive, complex cryptography. To be able to address the four challenges simultaneously in a more efficient manner, we design an aid-distribution system that does not require centralizing data to achieve the desired properties. Concretely, our contributions are the following:

- We propose a token-based aid-distribution system that is secure, privacy-preserving, auditable, and that can operate with little to no connectivity.
- We instantiate this system into two solutions that address the four challenges. Both solutions (i) ensure that even if multiple tokens have been assigned to the same household, a household can receive aid only once per distribution round (challenge 1) (ii) do not need to reveal any other information about households beyond their entitlement to request aid (challenge 4); and (iii) produce privacy-preserving audit proofs to guarantee auditability (challenge 3). The first solution addresses challenge 2 by using smart cards as tokens, and the second solution reduces cost by using recipients' smartphones as tokens when aid is distributed in areas where such devices are available.
- For each solution, we prove that the protocols we propose fulfill the security and privacy requirements of the ICRC, and we empirically demonstrate that they can operate at scale.
- We discuss deployment considerations required to bring secure and privacy-preserving digital solutions to the humanitarian setting.

2. Aid Distribution in Humanitarian Context

In this section, we describe how aid distribution works within the ICRC and elicit the requirements that a digitally supported aid-distribution system must fulfill.

2.1. Requirements Gathering

To understand the needs of the ICRC, we worked closely with staff from the ICRC Data Protection Office. We also organized two dedicated workshops with staff who are experienced in field operations, and held meetings with the staff in charge of organizing and coordinating aid-distribution programs. We refined the requirements in weekly meetings held with the Data Protection Office for more than a year.

Functional requirements. Humanitarian organizations require the following functionality for an aid-distribution system to be suitable in their context:

F1$_{household}$: Distribution per household. In some programs, humanitarian aid is allocated to individuals. However, we find that often the entitlement to aid is decided based on the *needs of households* [27]. To this end, at least one member of the household must register with the ICRC to collect aid (see requirement D3$_{robust}$ below). In this paper, we focus on per-household distribution, as any household-oriented solution can be trivially adapted to per-individual entitlement by equating households with individuals.

F2$_{modify}$: Entitlement modification after registration. In general, the ICRC expects the allocation to a household to not change during the aid-distribution program. Yet, aid recipients' needs may change through time, e.g., due to births, deaths, migration, or marriages. The ICRC must be able to modify the allocation to a household at any moment.

F3$_{periodic}$: Periodic distribution. In many cases, one-time distribution is enough to respond to an emergency. However, some aid-distribution programs are long-term, and the distribution of aid is divided into several *distribution periods* [33]. Recipients receive aid once per period (e.g., getting five bags of rice per month). In this situation, it is desirable that a household, through their representative(s), can get aid periodically without having to register every time.

Deployment requirements. Humanitarian organizations operate in extreme settings. These conditions constrain the aid-distribution system design space, as well as the technologies that can be used.

D1$_{low}$: Low-end hardware and sparse connectivity. The ICRC may operate in areas where neither high-end hardware nor stable connectivity is available. For example, recipients may not be able to afford high-end hardware, or the weather conditions may prevent the use of hardware that may be seen as a commodity in Western societies (e.g., in some locations the temperature or humidity can be too high for the latest models of smartphones). Therefore, aid-distribution systems should require only low-end hardware that is available in developing countries, be functional with little connectivity to the internet, and be reliable in austere environments.

D2$_{scale}$: Efficiency at medium scale. The ICRC distributes goods in unstable regions [27]. Letting a lot of recipients wait for a long time in such areas may put at risk these recipients, as well as the ICRC staff and collaborators, e.g., creating a high-value target for terrorism. In addition, queuing for basic human needs such as food may be questionable with respect to human dignity. Therefore, it is vital for the distribution system to be efficient even for numerous recipients. From our conversation with the ICRC, aid-distribution programs tend to involve thousands of recipients and, in some occasions, can involve more than 300,000 households per year.

D3$_{robust}$: Robust distribution. Getting aid is critical for the people in need, so the system must guarantee distribution even when unexpected events occur. The ICRC staff reports two common situations in which robustness is needed. First, some recipients lose or accidentally damage the proof of registration and entitlement (e.g., a voucher or an aid-distribution card). Second, registered recipients may not be able to attend the distribution (e.g., in case of sickness or travel). The system should provide means for recipients to receive a new proof or registration, and it is desirable that more than one household member can collect the goods.

D4$_{usability}$: Usability. Aid distribution programs often take place in locations where there is a lack of digital literacy.

This limits the type of solutions that humanitarian organizations can deploy, as not all technologies can be understood and used by the recipients.

We do not directly address this challenge in this work, but we choose smart cards and smartphone as platforms to develop our solution since both kind of devices have already been used by humanitarian organizations in the field.

Security requirements. Humanitarian organizations are funded by voluntary contributions and there is no guarantee that such contributions will continue long-term [34]. To ensure the continuity and the stability of fundraising, donors must trust the distribution system. To promote trust, aid-distribution systems must include mechanisms to ensure that aid is only distributed to their legitimate recipients in the allocated quantity.

$S1_{\text{limit}}$: Recipients should not be able to request more goods than they are entitled to. Legitimate recipients may try to get more aid than their household is entitled to (e.g., by requesting their entitlement more than once). To ensure that the limited resources of the ICRC can be used to aid as many people as possible, the aid-distribution system should guarantee that recipients cannot request more goods than those established by the criteria. We note that no technological solution can ensure that recipients do not request less goods (e.g., they may not attend the distribution).

$S2_{\text{legitimate}}$: Only legitimate recipients should be able to obtain aid. Illegitimate recipients may try to get aid, e.g., by trying to impersonate a legitimate recipient.

$S3_{\text{auditS}}$: Distribution must be auditable. To promote accountability of aid distribution programs, humanitarian organizations often keep track of their assistance activities by systematically collecting proof-of-delivery documents showing that goods are actually transferred from the organization to the recipient. Distribution tracking must enable auditors to find inconsistencies between the amounts distributed and the amounts delivered to eligible recipients in the field.

Privacy requirements. In the humanitarian context, recipients of aid rarely have any control or choice over which distribution system they use to get aid. Any leakage of recipient's personal information, during operation of the program, or afterwards if the humanitarian organization is forced to leave the area without cleaning up the distribution system, can can put the recipients in danger. For example, systems built by Western actors in Afghanistan collected sensitive biometric data. In August 2021, when those actors had to leave the country, these sensitive data were left in the hands of the Taliban, raising concerns that these biometrics could be used to target political opponents [26]. Hence, distribution systems in humanitarian settings must protect recipients' sensitive data to prevent any harm that could result from leaking these data.

$P1_{\text{registration}}$: Privacy at registration. Registration stations unavoidably learn some information about recipients to determine their eligibility to receive aid. Such information could include place of residence, household demographics, economic situations as well as identities. These necessary data cannot be hidden during registration. Yet, registration stations do not need to know transactional data of which recipients receive aid when and where.

$P2_{\text{distribution}}$: Privacy at distribution. Distribution stations must learn the eligibility and entitlement of a recipient in order to provide a recipient with the correct items. Yet, because distribution stations are often run by third parties, the stations should not learn any other information about recipients beyond what is required to provide the assistance.

$P3_{\text{auditP}}$: Privacy at auditing. Auditors receive audit logs to verify the correct functioning of distribution stations. Such logs may contain sensitive information about recipients (e.g., when or how often a specific recipient received their goods). To avoid endangering recipients, the system should ensure minimal disclosure of recipients' sensitive information to auditors. In most cases, it suffices to provide the total amount of distributed aid over a given time window and a proof that this aid was given to legitimate recipients.

$P4_{\text{biometrics}}$: Privacy of biometrics. Some aid-distribution systems use biometrics to ensure non-transferrability of aid. Biometric data are extremely privacy-sensitive. Because of privacy concerns, the ICRC's biometrics policy specifies that biometric templates must be stored on a device held by the data subject (i.e., aid recipient) whenever possible [31]. In particular, storing biometrics in central database held by the ICRC or service provider (e.g., registration or distribution stations) is considered as highly risky and must be avoided [31].

2.2. Humanitarian Aid Distribution Workflow

In this section, we provide a high-level description of a typical aid distribution workflow. We consider four actors whose interactions we illustrate in Fig. 1:

- **Recipients.** Recipients are those individuals that receive aid. We assume that recipients belong to one (and only one) household. We say an individual is a *legitimate* recipient for a household if they are registered for getting aid of that household; and we say they are *illegitimate* recipients if they are not registered in the system, or not associated to the household for which they receive aid.
- **Registration Station.** The registration station is in charge of enrolling legitimate recipients and determining their entitlement. Registration data are usually handled directly by the ICRC, or other trusted local parties. As such, the registration stations usually rely on the privileges and immunities afforded to the ICRC [6].
- **Distribution Station.** Distribution stations are in charge of distributing aid to legitimate recipients. Distribution stations are often operated by third parties.
- **Auditors.** Auditors are potentially external parties that use audit records produced by distribution station to verify the honest behavior of these stations.

Given an emergency situation, the ICRC works with local communities to set the criteria for selecting households that are eligible for receiving aid. Then, the organization registers the legitimate recipients for those households together

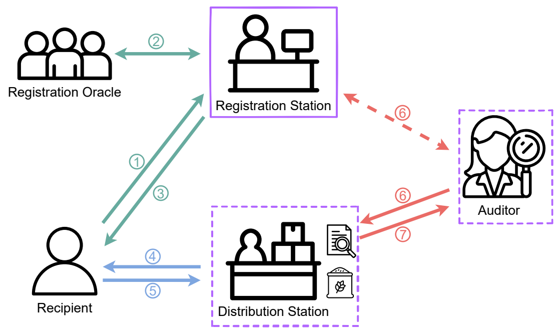

Figure 1. Workflow of humanitarian aid-distribution. Recipients interact with the registration station. The registration station relies on a registration oracle to assess correctness of the information. Recipients then request aid from the distribution station. The auditor can ask the distribution station for transaction records. The humanitarian organization controls the registration station, while the distribution station and the auditor can be under the control of third parties.

with their *entitlement* (e.g., five bags rice per month) [28] (Fig. 1, step ①). Registration can happen, among others, by setting up a registration desk in the emergency area, or by having the ICRC staff visit households and register their entitlement. Regardless of where registration happens, we will refer to the process as happening at the registration station. During registration, the registration station may need to validate the eligibility of households and their aid request, and that recipients are only registered for one household. This is typically done with the help of local communities (e.g., consulting with village elders) [27], [29], [32], [49], [52]. In this paper, we abstract the validation process as a registration oracle that can assign recipients to households and transfers this knowledge to the registration station (Fig. 1, step ②). After consulting the oracle, the registration station returns the result (e.g., a voucher, a card) to the recipient (Fig. 1, step ③).

Distribution of aid happens at *distribution stations*. These may share the same location as the registration station. In practice, there can be several registration and distribution stations. For simplicity, in this paper we abstract them as a single station and discuss how to extend our solution to multiple sites in Sect. 8.

Recipients request aid on behalf of the household at the distribution station (Fig. 1, steps ④ and ⑤). The distribution station verifies that recipients are legitimate and hands over the goods according to the recipient's entitlement.

When necessary, auditors can use information from the registration and distribution stations to verify that the aid distribution was done honestly (Fig. 1, steps ⑥ and ⑦).

Threat model. With respect to security of aid distribution ($S1_{\text{limit}}$, $S2_{\text{legitimate}}$) we assume that the registration and distribution stations are honest. This assumption is essential. A malicious registration station can simply register illegitimate recipients and change their entitlement, and a malicious distribution station could distribute however much aid to whomever it chooses. We assume (potentially illegitimate) recipients are malicious and aim to violate these security properties. For auditability ($S3_{\text{auditS}}$) we assume a malicious distribution station that tries to pass the audit check, but assume the registration station is honest.

With respect to privacy of registration data, we must trust the registration station (recall, these data are needed to verify eligibility). All other parties – distribution stations, auditors, and other recipients – are considered malicious. For privacy of transaction records and privacy of biometric data ($P4_{\text{biometrics}}$) we assume that all parties (except the recipient themselves) are malicious.

2.3. Existing Systems and Limitations

We review existing aid-distribution systems and evaluate to what extent they fulfill the above requirements.

Paper-based. The ICRC reports that, currently, most aid-distribution systems are paper-based. We identify two kinds of paper-based systems which differ in how the information used at distribution is stored.

In *pen-and-list* solutions, the information about eligible households, their members, and their entitlement, is stored in one *centralized* paper list during registration, and this list is passed to the distribution station. When recipients show up at the distribution station to claim their goods, the staff checks their identity to find them on the list. Recipients can sign their names as the evidence of distribution, producing a proof-of-delivery to be used for auditing purposes.

Pen-and-list solutions satisfy the functionality requirements ($F1_{\text{household}}$, $F2_{\text{modify}}$, $F3_{\text{periodic}}$). However, the verification of recipients at the distribution station does not scale ($D2_{\text{scale}}$). It can take long time to find a name among many others, especially when names are in languages for which the distribution staff are not native or when names are handwritten. In addition, if the list is acquired by third parties at registration, or during or after the distribution, *all* information about *all* registered participants would be revealed, breaching the privacy requirements ($P2_{\text{distribution}}$, $P3_{\text{auditP}}$).

In *pen-and-voucher* solutions, the information about eligibility, household membership, and entitlement is *decentralized* in vouchers given to the recipients. The distribution station takes as input the information from recipients and, upon verification, it delivers the goods. The staff at the distribution station can mark the voucher (e.g., by punching a hole) to show that the aid has been given in the corresponding month.

Compared to using the pen-and-list approach, which enables easy modification of entitlement, the decentralized nature of the pen-and-voucher solution means modification needs the assistance from recipients who may not be willing to cooperate ($F2_{\text{modify}}$). Moreover, because information about distribution is only recorded on the voucher, auditing would require to physically gather vouchers that would act as proof-of-delivery ($S3_{\text{auditS}}$). Gathering voucher for auditing can be difficult, e.g., when using punch-cards that are carried by recipients. Even when vouchers can be easily collected, there may be personal information written or printed on the voucher, e.g., to make sure only legitimate recipients can use them ($S2_{\text{legitimate}}$). This information may breach privacy at auditing ($P3_{\text{auditP}}$).

Digital solutions. Aiming to improve scalability and auditability of paper-based systems, humanitarian organizations are exploring digital solutions. Digital solutions are easy to scale and can easily provide detailed logs for auditing. Depending on their implementation they can have different drawbacks. Solutions that digitalize vouchers in a straightforward manner suffer from the same problem: anyone with a (digital) voucher can request the goods even though the voucher does not belong to this person ($S2_{\text{legitimate}}$). To solve this problem, the distribution station needs to authenticate the recipient. For this purpose, a lot of aid-distribution systems integrate Identity Management systems (IdM) [25] and strong authentication solutions such as biometrics. In this way, the distribution station can verify ownership before providing the goods. While this resolves the legitimacy requirement, IdM-based solutions do not fulfill the privacy requirements ($P1_{\text{registration}}$, $P2_{\text{distribution}}$, $P3_{\text{auditP}}$, $P4_{\text{biometrics}}$): recipients' behavior is linkable, the distribution station can identify recipients, auditing records can leak personal information, and the central biometric database becomes a single point of failure for privacy.

3. A Token-based Aid-distribution System

Next, we outline our proposal for a digital humanitarian aid-distribution system that fulfills the requirements from the ICRC as shown in Sect. 2.1.

3.1. Design choices

Decentralization of information into digital tokens. To avoid requiring high levels of trust on a single entity that collects all recipients' information, we store recipients' eligibility, entitlement information, and potentially authentication-oriented data on digital tokens that stay with the recipient. We trust tokens to keep this information private.

Using a token enables us to decentralize recipients' information. This in turn enables the design of a system in which the distribution station does not obtain information that allows to re-identify, or track the movements of, recipients ($P2_{\text{distribution}}$); and in which we can create auditing information without the need to trust the distribution station ($S3_{\text{auditS}}$). If the system requires the use of authentication information such as biometrics, these can be stored on the token, avoiding the creation of a centralized database ($P4_{\text{biometrics}}$). We discuss the use of biometrics in Sect. 8.

The use of *digital* tokens removes the need to manually check eligibility at distribution (e.g., finding names on a list or checking information on a paper voucher), increasing the efficiency of the system ($D2_{\text{scale}}$).

Enabling offline use of tokens. To fulfill $D1_{\text{low}}$, we designed our protocols to work offline. Tokens communicate locally with registration and distribution stations and do not require Internet access. Throughout this paper, we assume that the communication between tokens and stations is encrypted and authenticated. As this can be achieved using standard techniques on top of our protocols, e.g., during session establishment, we omit channel encryption and authentication from the description of our systems.

Modifying information via revocation and re-issuance. Modification of information in a decentralized system is cumbersome, as once information is stored on a recipient's token the registration station has no access to it. Since modifications are rare ($F2_{\text{modify}}$), we solve this problem by revoking tokens and re-issuing new ones with up-to-date information. We choose to implement blocklist-based revocation rather than allowlist-based revocation [11] because an allowlist-based method requires the knowledge of all the valid credentials, which may not be feasible in the field. In our design, the token receives the blocklist from the distribution station.

Preventing double dipping using household tags. A household must not be able to obtain aid more than once per round ($S1_{\text{limit}}$). At first thought, the use of strong authentication at distribution to detect recipients requesting aid more than once may seem to address this requirement. However, this does not prevent different members of the household from requesting the household-allocated aid. To ensure that different members of the same household can not obtain aid more than once per round, tokens output a household- and round-specific pseudorandom tag which they provide to the distribution station upon aid collection. The distribution station stores these tags. If another household member attempts to collect aid they must reveal their household tag for the round. Thus, the distribution station can easily detect the double-dipping attempt. We note that this mechanism does not require identifying recipients.

Non-forgeable auditing combined with crosschecking. Recall that to detect whether there have been goods not actually transferred from the organization to the recipient, the auditor should be able to find inconsistencies between the audit records and the warehouse storage. We model this by enabling the distribution station to prove the total amount of goods it was authorized to deliver to legitimate recipients. To prove this, we require tokens to output non-forgeable records at distribution that can be seen as proofs-of-delivery. These records can be aggregated, becoming a proof-of-delivery of the total amount of goods given out by the station. By comparing the aggregated total to the warehouse records, auditors can detect when the station distributes more goods than those that recipients asked for in the round (e.g., the station distributed also to illegitimate recipients or gave extra goods to legitimate recipients). When doing this comparison, auditors need to consider that numbers may not coincide for legitimate reasons (e.g., goods get broken before delivery, or some goods need to be distributed in an emergency with no time for proof collection). These cases cannot be addressed by technology and need to be resolved using analog means (e.g., with manual annotations) as in the current system. We also note that our auditing solution cannot detect the transfer of goods between legitimate recipients.

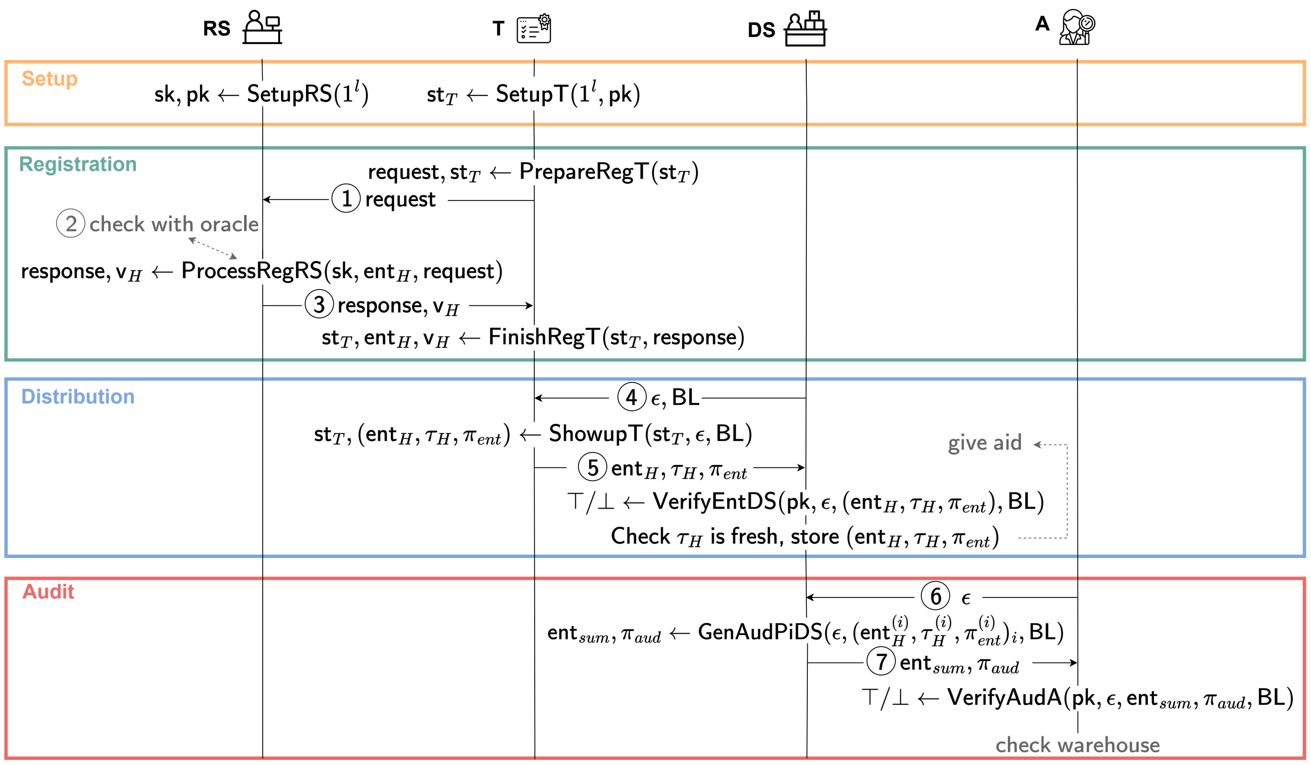
RS
T
DS
A
Setup
sk, pk ← SetupRS(1^l)
st_T ← SetupT(1^l, pk)
Registration
request, st_T ← PrepareRegT(st_T)
1 request
2 check with oracle
response, v_H ← ProcessRegRS(sk, ent_H, request)
3 response, v_H
st_T, ent_H, v_H ← FinishRegT(st_T, response)
Distribution
4 ϵ, BL
st_T, (ent_H, τ_H, π_{ent}) ← ShowupT(st_T, ϵ, BL)
give aid
5 ent_H, τ_H, π_{ent}
$\top/\bot$ ← VerifyEntDS(pk, ϵ, (ent_H, τ_H, π_{ent}), BL)
Check τ_H is fresh, store (ent_H, τ_H, π_{ent})
Audit
6 ϵ
ent_{sum}, π_{aud} ← GenAudPiDS(ϵ, $(\mathsf{ent}_H^{(i)}, \tau_H^{(i)}, \pi_{ent}^{(i)})_i$, BL)
7 ent_{sum}, π_{aud}
$\top/\bot$ ← VerifyAudA(pk, ϵ, ent_{sum}, π_{aud}, BL)
check warehouse

Figure 2. Overview of the token-based scheme omitting the global setup GlobalSetup. From left to right, there are four parties: Registration Station (RS), Token (T), Distribution Station (DS), and Auditor (A). The boxes from top to bottom represent four phases: setup phase, registration phase, distribution phase, and auditing phase. The circled numbers match steps from Fig. 1 to the scheme. Actions happening outside the scheme are marked in gray.

3.2. A Token-based Scheme

Our token-based scheme proceeds in four phases – setup, registration, distribution, and auditing – each consisting of several algorithms, whose syntax we describe below. Fig. 2 shows how parties use these algorithms in an aid-distribution system. If a party aborts while running an algorithm, every field of the output is set to $\bot$.

Setup Phase. To set up the system, a trusted party generates cryptographic parameters that are shared by all parties. The registration station generates a key pair (sk, pk). Tokens are stateful, and token-specific algorithms update an explicit state st_T that is initialized during setup. The algorithms satisfy the following syntax.

- param ← GlobalSetup(1^ℓ). A trusted party takes as input the security parameter 1^ℓ, and returns the public parameters param. The public parameters param are implicit inputs to all other algorithms. In practice, these parameters would be obtained from well-known recommendations by experts.
- sk, pk ← SetupRS(1^ℓ). The registration station takes as input the security parameter 1^ℓ, and outputs a private-public key pair (sk, pk). The public key pk is known to all parties.
- st_T ← SetupT(1^ℓ, pk). The token takes as input the security parameter 1^ℓ and the public key pk. The token returns an initial state st_T.

Registration Phase. Recipients use a token to register in the system. This token can either be provided to them (e.g., a smart card) or be brought by the recipient (e.g., a smartphone). To start registration, the token sends a registration request to the registration station together with a description of the household (Fig. 2, step ①). If the registration oracle agrees that this household is eligible and the recipient is a member of the household, the oracle outputs an entitlement ent_H (Fig. 2, step ②). The registration station computes the response and sends it back to the token (Fig. 2, step ③). The algorithms satisfy the following syntax.

- request, st_T ← PrepareRegT(st_T). The token takes as input the internal state st_T and outputs request information request with a new state st_T.
- response, v_H ← ProcessRegRS(sk, ent_H, request). The registration station takes as input a private key sk, the entitlement ent_H of the household, and the request request. It outputs a response response, which includes information about entitlement, and a revocation value v_H for this household.
- st_T, ent_H, v_H ← FinishRegT(st_T, response). The token finishes registration by taking as input the token state st_T and the registration response response. It returns a new state st_T, the entitlement ent_H, and the revocation value v_H. For simplicity, we assume that ent_H is a scalar, but our scheme can trivially extend to vectors to support different types of goods.

The registration station stores the mapping between households and corresponding revocation values v_H. To revoke a household's tokens, the registration station adds v_H to a public blocklist BL.

Distribution Phase. To receive aid, recipients go to the distribution station and start a request using their token. The station sends the current period ϵ and blocklist BL to the token (Fig. 2, step ④). The token ensures the period ϵ is not in the past and verifies that its own revocation value is not in BL. Then, the token sends the entitlement ent_H, a period-specific tag τ_H, and a proof of entitlement π_{ent} to the distribution station (Fig. 2, step ⑤). The station checks the proof of entitlement to verify that ent_H is correct and that the token has not been blocked (VerifyEntDS); and checks that the tag τ_H has not been seen before. If everything is correct, the station staff hands over the goods to the recipient, and the distribution station stores the tuple $(\mathsf{ent}_H, \tau_H, \pi_{\text{ent}})$ for auditing purposes. The algorithms satisfy the following syntax.

- $\mathsf{st}_T, (\mathsf{ent}_H, \tau_H, \pi_{\text{ent}}) \leftarrow \mathsf{ShowupT}(\mathsf{st}_T, \epsilon, \mathsf{BL})$. The token takes as input its own state st_T, the distribution period ϵ, and a blocklist BL. The algorithm outputs an updated state st_T, the entitlement ent_H, a tag τ_H that is unique for this household in this period, and a proof of the entitlement π_{ent}.
- $\top/\bot \leftarrow \mathsf{VerifyEntDS}(\mathsf{pk}, \epsilon, (\mathsf{ent}_H, \tau_H, \pi_{\text{ent}}), \mathsf{BL})$. The distribution station takes as input the public key pk, the period ϵ, the tuple $(\mathsf{ent}_H, \tau_H, \pi_{\text{ent}})$ from the token, and the blocklist BL. The algorithm outputs $\top$ if π_{ent} verifies, $\bot$ otherwise.

Audit Phase. To audit transactions, the auditor requests an audit proof. The distribution station computes the transaction total $\mathsf{ent}_{\text{sum}}$ as well as a proof π_{aud} and sends it to the auditor (GenAudPiDS and Fig. 2, step ⑦). The auditor uses the corresponding blocklist BL to verify the proof π_{aud} (VerifyAudA). The algorithms satisfy the following syntax.

- $\mathsf{ent}_{\text{sum}}, \pi_{\text{aud}} \leftarrow \mathsf{GenAudPiDS}(\epsilon, (\mathsf{ent}_H^{(i)}, \tau_H^{(i)}, \pi_{\text{ent}}^{(i)})_i, \mathsf{BL})$. The distribution station takes as input the period ϵ, tuples $(\mathsf{ent}_H^{(i)}, \tau_H^{(i)}, \pi_{\text{ent}}^{(i)})_i$, and a blocklist BL. It outputs the total entitlement $\mathsf{ent}_{\text{sum}}$ and an audit proof π_{aud}.
- $\top/\bot \leftarrow \mathsf{VerifyAudA}(\mathsf{pk}, \epsilon, \mathsf{ent}_{\text{sum}}, \pi_{\text{aud}}, \mathsf{BL})$. The auditor takes as input the period ϵ, the total entitlement $\mathsf{ent}_{\text{sum}}$, the audit proof π_{aud}, and the blocklist BL that was used for these records. The algorithm outputs $\top$ if π_{aud} verifies, $\bot$ otherwise.

4. Smart-card-based Aid Distribution

We propose the first instantiation of the design in Sect. 3 which uses smart cards as tokens.

Cheap, trustworthy tokens. Smart cards are low-cost enough that humanitarian organizations can give one or multiple cards to a household ($\mathrm{F1}_{\text{household}}$, $\mathrm{D1}_{\text{low}}$, $\mathrm{D3}_{\text{robust}}$). Yet, smart cards have enough storage and computation capacity to perform the cryptographic operations we need. And they can be treated as trusted execution environments (TEEs), leading to a simple design. Specifically, we assume attackers cannot modify or observe computations inside the card.

Enabling privacy-friendly auditing. Yet, the trusted nature of smart cards does not directly translate into privacy-friendly auditability ($\mathrm{S3}_{\text{auditS}}$): The distribution station must be able to convince the auditor that it only interacted with cards of legitimate receivers without violating recipient privacy. The key idea in our design is that during distribution the smart card signs a *homomorphic commitment to the entitlement*, and reveals its opening to the distribution station. The distribution station can compute a commitment to the total entitlement obtained by homomorphic addition of all individual commitments. The auditor can validate the signatures on the individual commitments, and check that the total entitlement matches the combined commitment, given an opening produced by the distribution station.

Enabling multiple cards per household. Preventing double dipping ($\mathrm{S1}_{\text{limit}}$) with a *single card* per household is easy – the (trusted) card can remember which periods it successfully completed. However, for $\mathrm{D3}_{\text{robust}}$ our system requires multiple cards per household. To enable the detection of double dipping, cards output a per-household per-round tag τ_H that they compute using a household secret k_H and the current round ϵ. This household secret needs to be present in all the cards of the household. We propose a *card-whispering* protocol that enables "cloning" of cards.

4.1. Preliminaries

We use a digital signature scheme given by the algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$. The key-generation algorithm $\mathsf{Gen}(1^\ell)$ takes as input a security parameter 1^ℓ and outputs a privacy-public key-pair $(\mathsf{sk}, \mathsf{pk})$. The signing algorithm $\mathsf{Sign}(\mathsf{sk}, m)$ takes as input the private key sk, a message m, and outputs a signature σ. The verification algorithm $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$ outputs $\top$ if σ is a valid signature on m and $\bot$ otherwise.

We denote by $\mathsf{PRF}_k : \{0,1\}^n \to \{0,1\}^n$ a pseudorandom function with key $k \in \{0,1\}^n$. We assume the output of this function is indistinguishable from a random function.

We use the additively-homomorphic Pedersen commitment scheme [46], defined by the algorithms $\mathsf{Com.Gen}$ and Commit. The function $\mathsf{Com.Gen}(1^\ell)$ takes as input the security parameter 1^ℓ and outputs parameters $\mathsf{param}_{PC} = (\mathbb{G}, q, g, h)$, where $\mathbb{G}$ is a cyclic group of prime order q generated by g, and h is another generator of $\mathbb{G}$ obtained by hashing a public string to a group element (this ensures that the discrete logarithm of h with respect to g is unknown to all parties). The function $\mathsf{Commit}(m, r)$ takes as input a message $m \in \mathbb{Z}_q$ and randomizer $r \in \mathbb{Z}_q$ and outputs a commitment $c = g^m h^r$. This commitment scheme is additively homomorphic: $\mathsf{Commit}(m_1, r_1) \cdot \mathsf{Commit}(m_2, r_2) = \mathsf{Commit}(m_1 + m_2, r_1 + r_2)$.

4.2. Smartcard-based Aid Distribution Protocol

We define the smart card system by instantiating the scheme from Sect. 3.2.

Setup. In the smart card system, the global setup (GlobalSetup) outputs public parameters for the Pedersen commitment scheme and a hash function H. The registration station generates a key pair $(\mathsf{sk}, \mathsf{pk})$. The card initiates a last-seen-period counter necessary for $\mathrm{P2}_{\text{distribution}}$ (see distribution phase below). The algorithms are implemented as follows:

- $\mathsf{param} \leftarrow \mathsf{GlobalSetup}(1^\ell)$. The global setup function runs $\mathsf{param}_{PC} \leftarrow \mathsf{Com.Gen}(1^\ell)$ to obtain Pedersen commitment parameters and picks a cryptographic hash function $H : \{0,1\}^* \rightarrow \{0,1\}^\ell$. It returns returns $\mathsf{param} = (\mathsf{param}_{PC}, H)$.

- $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{SetupRS}(1^\ell)$. The registration station takes as input 1^ℓ and runs $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\ell)$ to obtain a signing key pair.
- $\mathsf{st}_T \leftarrow \mathsf{SetupT}(1^\ell, \mathsf{pk})$. The token takes as input the security parameter 1^ℓ and the public key pk. Let the last-seen-period counter $\epsilon_{\text{last}} = 0$. The function returns the state $\mathsf{st}_T = (\epsilon_{\text{last}}, \mathsf{pk})$.

Registration. To register the first recipient of a household, the registration station collaborates with the registration oracle to check eligibility and determine the household's entitlement. (To register additional members for a household see the card-whispering section below.) The smart card does not send an initial $\mathsf{request}$ message, so we omit the definition of $\mathsf{PrepareRegT}$.

- $\mathsf{response}, \mathsf{v}_H \leftarrow \mathsf{ProcessRegRS}(\mathsf{sk}, \mathsf{ent}_H, \mathsf{request})$. On input of the private key sk and the entitlement ent_H ($\mathsf{request}$ is null), the registration station creates a random revocation value $\mathsf{v}_H \in_R \{0,1\}^{2\ell}$. It returns the response $\mathsf{response} = (\mathsf{sk}, \mathsf{ent}_H, \mathsf{v}_H)$.
- $\mathsf{st}_T, \mathsf{ent}_H, \mathsf{v}_H \leftarrow \mathsf{FinishRegT}(\mathsf{st}_T, \mathsf{response})$. On input of its state $\mathsf{st}_T = (\epsilon_{\text{last}}, \mathsf{pk})$ and the response $\mathsf{response}$ from the registration station, the card parses the response as $\mathsf{response} = (\mathsf{sk}, \mathsf{ent}_H, \mathsf{v}_H)$ and checks that the private key sk corresponds to the public key pk (as set during SetupT). The card generates a random household secret $\mathsf{k}_H \in_R \{0,1\}^\ell$ and returns the updated state $\mathsf{st}_T = (\epsilon_{\text{last}}, \mathsf{pk}, \mathsf{sk}, \mathsf{k}_H, \mathsf{ent}_H, \mathsf{v}_H)$.

Card-whispering. Recall that all cards assigned to the same household should be able to request aid (D3robust). To prevent double-dipping, all these cards must produce the same double-dipping tags (see distribution below). Thus, when registering members of an already registered household we clone a previous registered household member's card.

To preserve recipient privacy (P1registration, P2distribution), the system must limit when, and by whom this protocol can be run. Otherwise, an attacker with a cloned card can simply reproduce double-dipping tags τ_H and thus track a household's activities. We recommend that cloning can only happen *after* the card to be cloned successfully authenticates its owner (e.g., using biometrics, see Sect. 8) and *before* distribution starts. See Appendix D for the full protocol.

Distribution. We now describe how smart cards request aid, and how the distribution station verifies this request. Smart cards use the ϵ_{last} variable in their state to protect against malicious distribution stations by ensuring that they do not answer requests for past epochs (P2distribution).

- $\mathsf{st}_T, (\mathsf{ent}_H, \tau_H, \pi_{\text{ent}}) \leftarrow \mathsf{ShowupT}(\mathsf{st}_T, \epsilon, \mathsf{BL})$. The card takes as input its own state st_T, the period ϵ, and the latest blocklist BL provided by the distribution station. It parses st_T as $(\epsilon_{\text{last}}, \mathsf{pk}, \mathsf{sk}, \mathsf{k}_H, \mathsf{ent}_H, \mathsf{v}_H)$. The card checks that it has not been blocked and that the period ϵ is not in the past, i.e., $\mathsf{v}_H \notin \mathsf{BL} \wedge \epsilon_{\text{last}} < \epsilon$. If either of the checks fails, the card aborts. Otherwise, the card computes a tag τ_H, a commitment $\mathsf{Com}_{\text{ent}}$, and a signature σ_{aud},
$$\begin{aligned} \tau_H &= \mathsf{PRF}_{\mathsf{k}_H}(\epsilon) \\ \mathsf{Com}_{\text{ent}} &= \mathsf{Commit}(\mathsf{ent}_H, r), \quad r \in_R \mathbb{Z}_q \\ \sigma_{\text{aud}} &= \mathsf{Sign}(\mathsf{sk}, \tau_H \parallel \epsilon \parallel \mathsf{Com}_{\text{ent}} \parallel \mathsf{h}_{\text{BL}}), \end{aligned}$$
where $\mathsf{h}_{\text{BL}} = H(\mathsf{BL})$. Let $\pi_{\text{ent}} = (\sigma_{\text{aud}}, \mathsf{Com}_{\text{ent}}, r)$. It returns a new state st_T by updating ϵ_{last} to ϵ, and $(\mathsf{ent}_H, \tau_H, \pi_{\text{ent}})$.
- $\top/\bot \leftarrow \mathsf{VerifyEntDS}(\mathsf{pk}, \epsilon, (\mathsf{ent}_H, \tau_H, \pi_{\text{ent}}), \mathsf{BL})$. The distribution station parses the proof $\pi_{\text{ent}} = (\sigma_{\text{aud}}, \mathsf{Com}_{\text{ent}}, r)$ and checks the signature σ_{aud} by checking that $\mathsf{Verify}(\mathsf{pk}, \sigma_{\text{aud}}, \tau_H \parallel \epsilon \parallel \mathsf{Com}_{\text{ent}} \parallel \mathsf{h}_{\text{BL}}) = \top$ where $\mathsf{h}_{\text{BL}} = H(\mathsf{BL})$. It also verifies the commitment by checking that $\mathsf{Com}_{\text{ent}} = \mathsf{Commit}(\mathsf{ent}_H, r)$. If both checks pass, it returns $\top$, and $\bot$ otherwise.

The station additionally checks that it has not seen τ_H before handing out the goods (S1limit).

Auditing. The distribution station can use the recorded transactions $(\mathsf{ent}_H, \tau_H, \pi_{\text{ent}})$ to create audit proofs for the auditor. We assume all records are valid for ϵ and BL.

- $\mathsf{ent}_{\text{sum}}, , \pi_{\text{aud}} \leftarrow \mathsf{GenAudPiDS}(\epsilon, (\mathsf{ent}_H^{(i)}, \tau_H^{(i)}, \pi_{\text{ent}}^{(i)})_i, \mathsf{BL})$ The distribution station takes as input all the entries $(\mathsf{ent}_H^{(i)}, \tau_H^{(i)}, \pi_{\text{ent}}^{(i)})$ in a distribution period ϵ and lets
$$\pi_{\text{ent}}^{(i)} = (\sigma_{\text{aud}}^{(i)}, \epsilon, \mathsf{Com}_{\text{ent}}^{(i)}, r^{(i)}).$$
The distribution station computes the sum of entitlement and the sum of random values
$$\mathsf{ent}_{\text{sum}} = \sum_{i=1}^{n_\epsilon} \mathsf{ent}_H^{(i)}, \quad \mathsf{r}_{\text{sum}} = \sum_{i=1}^{n_\epsilon} r^{(i)}.$$
The station returns $\mathsf{ent}_{\text{sum}}$ together with an audit proof
$$\pi_{\text{aud}} = (\mathsf{r}_{\text{sum}}, (\sigma_{\text{aud}}^{(i)}, \tau_H^{(i)}, \mathsf{Com}_{\text{ent}}^{(i)})_i, \mathsf{h}_{\text{BL}}).$$
where $\mathsf{h}_{\text{BL}} = H(\mathsf{BL})$.
- $\top/\bot \leftarrow \mathsf{VerifyAudA}(\mathsf{pk}, \epsilon, \mathsf{ent}_{\text{sum}}, \pi_{\text{aud}}, \mathsf{BL})$. The auditor takes as input the public key pk, the period ϵ, the total entitlement $\mathsf{ent}_{\text{sum}}$, the auditing proof π_{aud}, and the blocklist BL. Let $\pi_{\text{aud}} = (\mathsf{r}_{\text{sum}}, (\sigma_{\text{aud}}^{(i)}, \tau_H^{(i)}, \mathsf{Com}_{\text{ent}}^{(i)})_i, \mathsf{h}_{\text{BL}})$. The auditor checks:
  - the validity of all signatures in the proof π_{aud}
$$\top = \mathsf{Verify}(\mathsf{pk}, \sigma_{\text{aud}}^{(i)}, \tau_H^{(i)} \parallel \epsilon \parallel \mathsf{Com}_{\text{ent}}^{(i)} \parallel \mathsf{h}_{\text{BL}}),$$
  - the uniqueness of all tags $\tau_H^{(i)}$
  - the sum of entitlement and the sum of random numbers match the commitment
$$\prod_{i=1}^{n_\epsilon} \mathsf{Com}_{\text{ent}}^{(i)} = \mathsf{Commit}(\mathsf{ent}_{\text{sum}}, \mathsf{r}_{\text{sum}})$$
  - that BL corresponds to h_{BL}, i.e., $\mathsf{h}_{\text{BL}} = H(\mathsf{BL})$.

  If all four checks pass, it outputs $\top$ and $\bot$ otherwise.

5. Smartphone Solution

In some areas where humanitarian organizations operate, smartphones are available (at least one per household). To take advantage of this fact, we propose a second instantiation of the design in Sect. 3 using smartphones as tokens.

Existing devices as tokens. Using smartphones as tokens reduces deployment costs associated with token distribution. And smartphones have much more computation and storage capacity than smart cards. However, unlike a smart card, smartphones can run arbitrary software. Thus, their output cannot be trusted. We address this by using advanced cryptography in three ways.

Proving eligibility. We use attribute-based credentials (ABCs) to enable the phone to prove the recipient's eligibility and entitlement to the distribution station. The use of ABCs lets us satisfy $\text{S2}_{\text{legitimate}}$ while, at the same time, ensuring privacy $\text{P2}_{\text{distribution}}$

We opt for the pairing-based ABC scheme by Pointcheval and Sanders (PS) [47]. We considered the use of other ABC schemes, but found them to be less suited for our purposes. Some schemes can only be shown once while maintaining unlinkability, and would thus require issuing one credential per distribution round [3], [8]; others require the issuer and the verifier to share keys, which would harm auditability in our setting [13]. There are alternatives offering the same functionality as PS credentials with no less complexity [2], [11].

Other technologies in previous research, e.g., FIDO [19] and PrivacyPass [16], are not suitable in our use case. To be more specific, FIDO does not provide unlinkability towards the same relying party (the distribution station), i.e., whenever the recipient shows up twice at the same distribution station, the actions are linkable. PrivacyPass could be used instead of ABCs by encoding entitlement into PrivacyPass tokens that are provided to the distribution station. However, it is hard to see how privacy pass tokens can support privacy-friendly auditability (which requires proving how many tokens were redeemed); or revocation (which requires privacy-friendly blocklisting).

Double-dipping tags. Instead of using a PRF to compute a double-dipping tag, we use a trick from direct anonymous attestation [9]: we compute $\tau_H = H(\epsilon)^{\mathsf{k}_H}$ where k_H is the household secret and $H : \{0,1\}^* \to \mathbb{G}$ is a cryptographic hash function mapping strings to group elements. We extend the zero-knowledge proof of having a credential to prove the correct computation of τ_H.

Proving non-revocation. Finally, we rely on cryptography to let smartphones prove that their credential has not been revoked. Since we aim to provide privacy during distribution against malicious registration and distribution stations, not all revocation mechanisms are appropriate (see, e.g., Lapon et al. [39] for an overview). Because the revoking party, i.e., the registration station, is not trusted, phones should be able to detect whether they have been revoked. This rules out approaches such as verifier local revocation [7] as well as any approach based on verifiable encryption.

Instead, we use anonymous blocklisting [24], [54] to let the phone prove that it is not currently revoked. These proofs are linear in the size of the blocklist. We considered dynamic accumulators [15], [40], [44] – whose non-revocation proofs are constant size – but decided against them due to their increased cryptographic complexity and the need to handle witness updates in a privacy-friendly manner.

5.1. Preliminaries: Attribute-based Credentials

We use ABCs as a building block, and describe them only at a high level. During the issuance protocol, between a user and an issuer, the issuer provides a signed credential to the user. A credential contains one or more attributes, some of which can be hidden from the issuer during the signing process. We use $C(\mathsf{k}_H, \mathsf{ent}_H, r_H)$ to denote a credential with the three attributes k_H, ent_H and r_H. Users can create a zero-knowledge proof of possessing a valid credential. For simplicity, we slightly abuse the notation and write:

$$\mathsf{NIZK}\{(C, \mathsf{k}_H, \mathsf{ent}_H, r_H) : C(\mathsf{k}_H, \mathsf{ent}_H, r_H)\}$$

to denote the non-interactive zero-knowledge proof of knowing a valid credential C with attributes $\mathsf{k}_H, \mathsf{ent}_H, r_H$. We refer to Appendix B.3 for more details about ABCs.

5.2. Smartphone-based Aid Distribution Protocol

We define the smartphone-based system by instantiating the scheme from Sect. 3.2.

Setup. The $\mathsf{GlobalSetup}$ function determines global parameters, including those for the ABC scheme. The registration station acts as the issuer in the ABC scheme, and thus, creates a key pair for it.

- $\mathsf{param} \leftarrow \mathsf{GlobalSetup}(1^\ell)$. The $\mathsf{GlobalSetup}$ algorithm takes as input the security parameter 1^ℓ. It computes three types of parameters. First, it generates for the ABC scheme a type-III bilinear group pair given by $\mathsf{param}_{\mathsf{PS}} = (e(\cdot,\cdot), \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g_1, g_2, g_T)$ where g_1, g_2, g_T are generators of the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order q respectively. Second, it sets up the Pedersen commitment scheme in $\mathbb{G}_1$ with parameters $\mathsf{param}_{PC} = (\mathbb{G}_1, q, g_1, h)$. Third, it picks a hash function $H : \{0,1\}^* \to \mathbb{G}_1$ that maps strings onto the group $\mathbb{G}_1$. The application provider publishes all the global parameters $\mathsf{param} = (\mathsf{param}_{\mathsf{PS}}, \mathsf{param}_{PC}, H)$.
- $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{SetupRS}(1^\ell)$. The registration station generates a signing key pair $(\mathsf{sk}, \mathsf{pk})$ for the ABC scheme.
- $\mathsf{st}_T \leftarrow \mathsf{SetupT}(1^\ell, \mathsf{pk})$. The phone takes as input 1^ℓ and pk from the registration station. It initiates the last-seen-period counter ϵ_{last} and phone state $\mathsf{st}_T = (\epsilon_{\text{last}}, \mathsf{pk})$.

It is essential for privacy that all phones receive the same public key pk. In the following, we assume this is the case.

Registration. Recipients take their smartphones when registering at the registration station. The station associates legitimate recipients to the household with the help from the registration oracle. The station runs the ABC issuance protocol jointly with the phone to issue a credential.

- $\mathsf{request}, \mathsf{st}_T \leftarrow \mathsf{PrepareRegT}(\mathsf{st}_T)$. The phone takes as input the initial state st_T which it parses as $(\epsilon_{\text{last}}, \mathsf{pk})$. It generates a household secret $\mathsf{k}_H \in_R \mathbb{Z}_q$. To enable revocation using anonymous blocklisting, the phone picks $r_H \in_R \mathbb{Z}_q$ and computes the revocation value $\mathsf{v}_H = (g_1, g_1^{r_H})$. The phone creates the first message $\mathsf{request}_{abc}$ in the ABC issuance protocol where k_H and r_H are phone-defined attributes. We assume that $\mathsf{request}_{\text{abc}}$ includes a proof that v_H is well formed (see Section B.3.2). The attribute k_H and r_H will be hidden from the registration station (i.e., the issuer). Let st_{iss} be the phone's private issuance state, $\mathsf{st}_T = (\epsilon_{\text{last}}, \mathsf{pk}, \mathsf{k}_H, r_H, \mathsf{st}_{\text{iss}})$ the updated token state. The phone returns $\mathsf{request} = (\mathsf{request}_{\text{abc}}, \mathsf{v}_H)$ and st_T.
- $\mathsf{response}, \mathsf{v}_H \leftarrow \mathsf{ProcessRegRS}(\mathsf{sk}, \mathsf{ent}_H, \mathsf{request})$. On input of its private key sk, the entitlement ent_H and the issue request $\mathsf{request} = (\mathsf{request}_{\text{abc}}, \mathsf{v}_H)$ from the phone, the registration station checks the validity of the request and that v_H has been correctly formed. Then it uses sk and the phone's request $\mathsf{request}_{\text{abc}}$ to create a preliminary credential on the attributes $(\mathsf{k}_H, \mathsf{ent}_H, r_H)$ (without learning either k_H or r_H) which it packs into $\mathsf{response}$. It returns $\mathsf{response}$ and v_H.
- $\mathsf{st}_T, \mathsf{ent}_H, \mathsf{v}_H \leftarrow \mathsf{FinishRegT}(\mathsf{st}_T, \mathsf{response})$. On input of its internal state $\mathsf{st}_T = (\epsilon_{\text{last}}, \mathsf{pk}, \mathsf{k}_H, r_H, \mathsf{st}_{\text{iss}})$ and the registration station's response $\mathsf{response}$, the phone completes the credential issuance protocol to obtain a credential C on the attributes $(\mathsf{k}_H, \mathsf{ent}_H, r_H)$ using its issuance state st_{iss} and $\mathsf{response}$. Then, it verifies that C is a valid credential under pk, and aborts otherwise. The phone returns the new state $\mathsf{st}_T = (\epsilon_{\text{last}}, \mathsf{pk}, C, \mathsf{k}_H, \mathsf{ent}_H, r_H)$, ent_H, and $\mathsf{v}_H = (g_1, g_1^{r_H})$.

Distribution. We next describe the algorithms used during aid collection.

- $\mathsf{st}_T, (\mathsf{ent}_H, \tau_H, \pi_{\text{ent}}) \leftarrow \mathsf{ShowupT}(\mathsf{st}_T, \epsilon, \mathsf{BL})$. The phone takes as input st_T, the period ϵ, and the blocklist BL. Let $\mathsf{st}_T = (\epsilon_{\text{last}}, \mathsf{pk}, C, \mathsf{k}_H, \mathsf{ent}_H, r_H)$. The phone verifies that ϵ is not in the past, and that it has not made a distribution request in this epoch before, i.e., it checks that $\epsilon_{\text{last}} < \epsilon$. Then, the phone checks that it is not blocked, i.e., that for all $(h, H) \in \mathsf{BL}$, $H \neq h^{r_H}$. If any check fails, the phone aborts. To continue, the phone computes a tag τ_H and a commitment $\mathsf{Com}_{\text{ent}}$:

$$\begin{aligned} \tau_H &= H(\epsilon)^{\mathsf{k}_H}, \\ \mathsf{Com}_{\text{ent}} &= \mathsf{Commit}(\mathsf{param}_{PC}, \mathsf{ent}_H, r), r \in_R \mathbb{Z}_q. \end{aligned}$$

  Finally, the phone constructs the proof

$$\begin{aligned} \pi_s = \mathsf{NIZK}\big\{(C, \mathsf{k}_H, \mathsf{ent}_H, r_H, r) : \tau_H = H(\epsilon)^{\mathsf{k}_H} \wedge \\ C(\mathsf{k}_H, \mathsf{ent}_H, r_H) \quad \wedge \quad \mathsf{Com}_{\text{ent}} = \mathsf{Commit}(\mathsf{ent}_H, r) \\ \wedge \quad \forall (h, H) \in \mathsf{BL} : H \neq h^{r_H}\big\}, \end{aligned}$$

  to show that it knows a valid credential C on $\mathsf{k}_H, \mathsf{ent}_H, r_H$, that τ_H is correctly constructed, that $\mathsf{Com}_{\text{ent}}$ is correctly computed, and that it is not revoked (using the blocklisting protocol [24]). Let $\mathsf{st}_T = (\epsilon, \mathsf{pk}, C, \mathsf{k}_H, \mathsf{ent}_H, r_H)$ be the new state and $\pi_{\text{ent}} = (\pi_s, \mathsf{Com}_{\text{ent}}, r)$. It returns st_T and $(\mathsf{ent}_H, \tau_H, \pi_{\text{ent}})$.
- $\top/\bot \leftarrow \mathsf{VerifyEntDS}(\mathsf{pk}, \epsilon, (\mathsf{ent}_H, \tau_H, \pi_{\text{ent}}), \mathsf{BL})$. On input of the public key pk, the period ϵ, the tuple $(\mathsf{ent}_H, \tau_H, \pi_{\text{ent}})$, and the blocklist BL, the distribution station proceeds as follows. First, it parses π_{ent} as $(\pi_s, \mathsf{Com}_{\text{ent}}, r)$. Then, it verifies the proof π_s using the public key pk and the provided blocklist BL. If the proof verifies, it returns $\top$, and $\bot$ otherwise.

Auditing. Auditing proceeds similarly to the smart card system, except that the auditor checks zero-knowledge proofs. Again, we assume the records are valid for ϵ and BL.

- $\mathsf{ent}_{\text{sum}}, \pi_{\text{aud}} \leftarrow \mathsf{GenAudPiDS}(\epsilon, (\mathsf{ent}_H^{(i)}, \tau_H^{(i)}, \pi_{\text{ent}}^{(i)})_i, \mathsf{BL})$. The distribution station takes as input all the entries $(\mathsf{ent}_H^{(i)}, \tau_H^{(i)}, \pi_{\text{ent}}^{(i)})$ in a distribution period ϵ and lets

$$\pi_{\text{ent}}^{(i)} = (\pi_s^{(i)}, \mathsf{Com}_{\text{ent}}^{(i)}, r^{(i)}).$$

  The distribution station computes the sum of entitlement $\mathsf{ent}_{\text{sum}} = \sum_{i=1}^{n_\epsilon} \mathsf{ent}_H^{(i)}$ and the randomizer $\mathsf{r}_{\text{sum}} = \sum_{i=1}^{n_\epsilon} r^{(i)}$. It returns $\mathsf{ent}_{\text{sum}}$ and an audit proof

$$\pi_{\text{aud}} = (\mathsf{r}_{\text{sum}}, (\pi_s^{(i)}, \tau_H^{(i)}, \mathsf{Com}_{\text{ent}}^{(i)})_i).$$

- $\top/\bot \leftarrow \mathsf{VerifyAudA}(\mathsf{pk}, \epsilon, \mathsf{ent}_{\text{sum}}, \pi_{\text{aud}}, \mathsf{BL})$. The auditor takes as input the period ϵ, the total entitlement $\mathsf{ent}_{\text{sum}}$, the auditing proof π_{aud}, and the blocklist BL that was used in ϵ. Let $\pi_{\text{aud}} = (\mathsf{r}_{\text{sum}}, (\pi_s^{(i)}, \tau_H^{(i)}, \mathsf{Com}_{\text{ent}}^{(i)})_i)$. The auditor checks the validity of all the proofs $\pi_s^{(i)}$ with respect to $\tau_H^{(i)}$, $\mathsf{Com}_{\text{ent}}^{(i)}$ and the blocklist BL; checks the uniqueness of all tags $\tau_H^{(i)}$; and checks that the sum of entitlement and the sum of random numbers match the commitment: $\prod_i \mathsf{Com}_{\text{ent}}^{(i)} = \mathsf{Commit}(\mathsf{ent}_{\text{sum}}, \mathsf{r}_{\text{sum}})$. If all checks pass, it outputs $\top$, and $\bot$ otherwise.

6. Properties and Proofs

We formalize the security and privacy properties needed to fulfill the requirements described in Sect. 2.1 using cryptographic games. In the games, the adversary interacts with users and parties in the system using oracles. See Algorithms 1 and 2 for an overview of the oracles we use. Oracles track information, e.g., the entitlement and revocation values assigned to recipients (in $\mathsf{Ent}, \mathsf{Rev}$), the honest and malicious recipients (in $\mathcal{H}, \mathcal{M}$), as well as the last epoch in which an honest recipient sent information to a distribution station (using ϵ_{last}). We use these to define win conditions.

In this section, we use proof sketches to show our two solutions fulfill the properties and refer to the full version of the paper for the complete proofs [57].

6.1. Privacy of Distribution

An aid-distribution system should preserve the privacy of recipients at distribution against (potentially malicious) registration and distribution stations (P1$_{\text{registration}}$, P2$_{\text{distribution}}$).

Algorithm 1 Registration Oracles

```
1: function $\mathcal{O}_{\text{HonestReg}}(\mathsf{id}, \mathsf{v}_H, \mathsf{ent}_H)$
2:     if $\mathsf{id} \in \mathcal{H} \cup \mathcal{M}$ then return $\perp$
3:     $\mathsf{st}_T^{(\mathsf{id})} \leftarrow \mathsf{SetupT}(1^\ell, \mathsf{pk})$
4:     $\mathsf{request}, \mathsf{st}_T^{(\mathsf{id})} \leftarrow \mathsf{PrepareRegT}(\mathsf{st}_T^{(\mathsf{id})})$
5:     $\mathsf{response}, \mathsf{v}_H \leftarrow \mathsf{ProcessRegRS}(\mathsf{sk}, \mathsf{ent}_H, \mathsf{request})$
6:     $\mathsf{st}_T^{(\mathsf{id})}, \mathsf{ent}_H, \mathsf{v}_H \leftarrow \mathsf{FinishRegT}(\mathsf{st}_T^{(\mathsf{id})}, \mathsf{response})$
7:     $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathsf{id}\}$, $\mathsf{Ent}[\mathsf{id}] \leftarrow \mathsf{ent}_H$, $\mathsf{Rev}[\mathsf{id}] \leftarrow \mathsf{v}_H$
8:     return $\mathsf{v}_H$
9: function $\mathcal{O}_{\text{MalUserReg}}(\mathsf{id}, \mathsf{ent}_H, \mathsf{request})$
10:    if $\mathsf{id} \in \mathcal{H} \cup \mathcal{M}$ then return $\perp$
11:    $\mathsf{response}, \mathsf{v}_H \leftarrow \mathsf{ProcessRegRS}(\mathsf{sk}, \mathsf{ent}_H, \mathsf{request})$
12:    $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mathsf{id}\}$, $\mathsf{Ent}[\mathsf{id}] \leftarrow \mathsf{ent}_H$, $\mathsf{Rev}[\mathsf{id}] \leftarrow \mathsf{v}_H$
13:    return response
14: function $\mathcal{O}_{\text{PrepareReg}}(\mathsf{id})$
15:    if $\mathsf{id} \in \mathcal{H}_{pre} \cup \mathcal{H}$ then return $\perp$
16:    $\mathsf{st}_T^{(\mathsf{id})} \leftarrow \mathsf{SetupT}(1^\ell, \mathsf{pk})$
17:    $\mathsf{request}, \mathsf{st}_T^{(\mathsf{id})} \leftarrow \mathsf{PrepareRegT}(\mathsf{st}_T^{(\mathsf{id})})$
18:    $\mathcal{H}_{pre} \leftarrow \mathcal{H}_{pre} \cup \{\mathsf{id}\}$
19:    return request
20: function $\mathcal{O}_{\text{FinishReg}}(\mathsf{id}, \mathsf{response})$
21:    if $\mathsf{id} \notin \mathcal{H}_{pre} \vee \mathsf{id} \in \mathcal{H}$ return $\perp$
22:    $\mathsf{st}_T^{(\mathsf{id})}, \mathsf{ent}_H, \mathsf{v}_H \leftarrow \mathsf{FinishRegT}(\mathsf{st}_T^{(\mathsf{id})}, \mathsf{response})$
23:    $\mathcal{H} \leftarrow \mathcal{H} \cup \{\mathsf{id}\}$, $\mathsf{Ent}[\mathsf{id}] \leftarrow \mathsf{ent}_H$, $\mathsf{Rev}[\mathsf{id}] \leftarrow \mathsf{v}_H$
```

Algorithm 2 Distribution Oracles

```
1: function $\mathcal{O}_{\text{Showup}}(\mathsf{id}, \epsilon, \mathsf{BL})$
2:     if $\mathsf{id} \notin \mathcal{H}$ return $\perp$
3:     if $\mathsf{Rev}[\mathsf{id}] \notin \mathsf{BL}$ then $\mathsf{ent}_{\text{sum}}[\epsilon, \mathsf{BL}] \leftarrow \mathsf{ent}_{\text{sum}}[\epsilon, \mathsf{BL}] + \mathsf{Ent}[\mathsf{id}]$
4:     $\epsilon^{\text{last}}[\mathsf{id}] \leftarrow \max(\epsilon^{\text{last}}[\mathsf{id}], \epsilon)$
5:     $\mathsf{st}_T^{(\mathsf{id})}, (\mathsf{ent}_H, \tau_H, \pi_{\text{ent}}) \leftarrow \mathsf{ShowupT}(\mathsf{st}_T^{(\mathsf{id})}, \epsilon, \mathsf{BL})$
6:     return $\mathsf{ent}_H$, $\tau_H$, $\pi_{\text{ent}}$
7: function $\mathcal{O}_{\text{ShowupTwo}}(\mathsf{id}_0, \mathsf{id}_1, \epsilon, \mathsf{BL})$
8:     if $\mathsf{id}_0, \mathsf{id}_1 \notin \mathcal{H}$ return $\perp$
9:     $\mathsf{BlockLists}[\epsilon] \leftarrow \mathsf{BlockLists}[\epsilon] \cup \{\mathsf{BL}\}$
10:    $\mathsf{st}_T^{(\mathsf{id}_0)}, (\mathsf{ent}_H^{(0)}, \tau_H^{(0)}, \pi_{\text{ent}}^{(0)}) \leftarrow \mathsf{ShowupT}(\mathsf{st}_T^{(\mathsf{id}_0)}, \epsilon, \mathsf{BL})$
11:    if $\mathsf{VerifyEntDS}(\mathsf{pk}, \epsilon, (\mathsf{ent}_H^{(0)}, \tau_H^{(0)}, \pi_{\text{ent}}^{(0)}), \mathsf{BL}) \wedge \tau_H^{(0)} \notin \mathsf{log}^0$
       then
12:        $\mathsf{log}^0[\epsilon] \leftarrow \mathsf{log}^0[\epsilon] \cup \{(\mathsf{ent}_H^{(0)}, \tau_H^{(0)}, \pi_{\text{ent}}^{(0)})\}$
13:    $\mathsf{st}_T^{(\mathsf{id}_1)}, (\mathsf{ent}_H^{(1)}, \tau_H^{(1)}, \pi_{\text{ent}}^{(1)}) \leftarrow \mathsf{ShowupT}(\mathsf{st}_T^{(\mathsf{id}_1)}, \epsilon, \mathsf{BL})$
14:    if $\mathsf{VerifyEntDS}(\mathsf{pk}, \epsilon, (\mathsf{ent}_H^{(1)}, \tau_H^{(1)}, \pi_{\text{ent}}^{(1)}), \mathsf{BL}) \wedge \tau_H^{(1)} \notin \mathsf{log}^1$
       then
15:        $\mathsf{log}^1[\epsilon] \leftarrow \mathsf{log}^1[\epsilon] \cup \{(\mathsf{ent}_H^{(1)}, \tau_H^{(1)}, \pi_{\text{ent}}^{(1)})\}$
```

We model this using the indistinguishability experiment in Algorithm 3. Adversary $\mathcal{A}$ plays the role of a malicious registration station and produces the corresponding public key pk (line 3). The adversary can use the oracles $\mathcal{O}_{\mathsf{PrepareReg}}$ and $\mathcal{O}_{\mathsf{FinishReg}}$ to interact as a (malicious) registration station with honest users (line 4). It can also act as a (malicious) distribution station and use $\mathcal{O}_{\mathsf{Showup}}$ to request that an honest user runs ShowupT. The oracle keeps track of the last epoch for which $\mathcal{O}_{\mathsf{Showup}}$ has been called for each user (line 4, Algorithm 2). The modeling assumes that all tokens receive the same public key during SetupT. In a real deployment, this assumption must be realized, e.g., by a trusted manufacturer (for cards) or application provider (for phones).

Algorithm 3 Indistinguishability experiment

```
1: function $\mathsf{Exp}_{\mathcal{A},b}^{\text{IND}}(1^\ell)$
2:     $\mathsf{param} \leftarrow \mathsf{GlobalSetup}(1^\ell)$
3:     $\mathsf{pk} \leftarrow \mathcal{A}(\mathsf{param})$
4:     $\mathsf{id}_0, \mathsf{id}_1, \epsilon^*, \mathsf{BL}^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{PrepareReg}}(\cdot), \mathcal{O}_{\mathsf{FinishReg}}(\cdot), \mathcal{O}_{\mathsf{Showup}}(\cdot)}()$
5:     $\epsilon_0^l = \epsilon^{\text{last}}[\mathsf{id}_0]$, $\epsilon_1^l = \epsilon^{\text{last}}[\mathsf{id}_1]$
6:     if $\mathsf{id}_0, \mathsf{id}_1 \notin \mathcal{H}$ then return $\perp$
7:     if     $\mathsf{Ent}[\mathsf{id}_0] \neq \mathsf{Ent}[\mathsf{id}_1]$ or
              $|\{\mathsf{Rev}[\mathsf{id}_0], \mathsf{Rev}[\mathsf{id}_1]\} \cap \mathsf{BL}^*| = 1$ or
              $(\epsilon^* > \min(\epsilon_0^l, \epsilon_1^l) \wedge \epsilon^* \leq \max(\epsilon_0^l, \epsilon_1^l))$ then
8:         return $\perp$
9:     $\mathsf{ent}_H^b, \tau_H^b, \pi_{\text{ent}}^b \leftarrow \mathcal{O}_{\mathsf{Showup}}(\mathsf{id}_b, \epsilon^*, \mathsf{BL}^*)$
10:    $b' \leftarrow \mathcal{A}(\mathsf{ent}_H^b, \tau_H^b, \pi_{\text{ent}}^b)$
11:    return $b' = b$
```

Eventually the adversary outputs two honest challenge users $\mathsf{id}_0, \mathsf{id}_1$, a challenge epoch ϵ^* and a challenge blocklist BL^* (line 4). The adversary's goal is to recognize one of these users during distribution. We rule out 3 trivial win conditions (line 7): (1) the selected recipients have different entitlements, (2) exactly one of the recipients is revoked, (3) exactly one of the tokens will run ShowupT in the selected period ϵ^*. If the adversary can determine the challenge bit b (line 11), it wins the game.

Definition 1. An aid-distribution system provides indistinguishability if the following advantage is negligible:

$$\mathsf{Adv}_{\mathcal{A}}^{\text{IND}} = \left|\Pr\left[\mathsf{Exp}_{\mathcal{A},0}^{\text{IND}}(1^\ell)\right] - \Pr\left[\mathsf{Exp}_{\mathcal{A},1}^{\text{IND}}(1^\ell)\right]\right|.$$

Theorem 1. *The smart card system has indistinguishability provided that PRF is a pseudo-random function.*

Proof sketch. We focus on the interesting case where both tokens produce a real output (i.e., they are not blocked and have not been in this epoch before). The response of the challenge user consists of $(\mathsf{ent}_H, \tau_H, \pi_{\text{ent}})$ where $\pi_{\text{ent}} = (\sigma_{\text{aud}}, \mathsf{Com}_{\text{ent}}, r)$. Because the entitlements of the challenge users are the same, we focus our attention on τ_H and σ_{aud}. Since all tokens share the same signing key, the signature σ_{aud} does not help in distinguishing tokens. Finally, since PRF is a pseudo-random function the adversary cannot use observations for challenge users in earlier epochs to recognize users. □

Theorem 2. *The smartphone system has indistinguishability in the random oracle model provided that the DDH assumption holds, the ABC scheme is unlinkable, and the blocklisting scheme is anonymous.*

Proof sketch. We focus on the interesting case where there is a response: $(\mathsf{ent}_H, \tau_H, \pi_{\text{ent}})$, with $\pi_{\text{ent}} = (\pi_s, \mathsf{Com}_{\text{ent}}, r)$. As the entitlements of the challenge users are the same, we focus on τ_H and π_S. First, by unlinkability of the ABC scheme and anonymity of the blocklisting scheme, we can simulate the proof π_s for the challenge users. What remains to show is that $\tau_H = H(\epsilon)^{\mathsf{k}_H}$ does not make households distinguishable. However, one can show that under the random oracle model and the DDH assumption, no adversary can distinguish $\tau_H^0 = H(\epsilon^*)^{\mathsf{k}_H^0}$ from $\tau_H^1 = H(\epsilon^*)^{\mathsf{k}_H^1}$ □

Algorithm 4 Auditability experiment

```
1: function $\mathsf{Exp}^{\mathrm{AUD}}_{\mathcal{A}}(1^\ell)$
2:     $\mathsf{param} \leftarrow \mathsf{GlobalSetup}(1^\ell)$
3:     $\mathsf{sk}, \mathsf{pk} \leftarrow \mathsf{SetupRS}(1^\ell)$
4:     $\epsilon^*, \mathsf{ent}^*_{\mathsf{sum}}, \pi^*_{\mathsf{aud}}, \mathsf{BL}^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{HonestReg}}(\cdot), \mathcal{O}_{\mathsf{MalUserReg}}(\cdot), \mathcal{O}_{\mathsf{Showup}}(\cdot)}(\mathsf{pk})$
5:     $\mathsf{valid} \leftarrow \mathsf{VerifyAudA}(\mathsf{pk}, \epsilon^*, \mathsf{ent}^*_{\mathsf{sum}}, \pi^*_{\mathsf{aud}}, \mathsf{BL}^*)$
6:     $\mathcal{R}_{\mathsf{mal}} = \{\mathsf{id} \mid \mathsf{Rev}[\mathsf{id}] \in \mathsf{BL}^* \wedge \mathsf{id} \in \mathcal{M}\}$
7:     $\mathsf{ent}_{\max} \leftarrow \mathsf{ent}_{\mathsf{sum}}[\epsilon^*, \mathsf{BL}^*] + \sum_{\mathsf{id} \in \mathcal{M} \setminus \mathcal{R}_{\mathsf{mal}}} \mathsf{Ent}[\mathsf{id}]$
8:     return $\mathsf{valid} \wedge (\mathsf{ent}^*_{\mathsf{sum}} > \mathsf{ent}_{max})$
```

6.2. Auditability

Aid-distribution systems should be auditable (S3$_{\text{auditS}}$): a malicious distribution station should not be able to convince an auditor that it provided more aid than what corresponds to the legitimate recipients that showed up. Since a malicious registration station could simply enroll more recipients as legitimate, making auditing moot, we assume the registration station is honest. We model auditability using the experiment in Algorithm 4. The adversary can register honest recipients using $\mathcal{O}_{\mathsf{HonestReg}}$. The oracle returns recipients' revocation value to enable blocklisting of those recipients. To register malicious recipients, the adversary uses $\mathcal{O}_{\mathsf{MalUserReg}}$. We model the fact that smart cards are TEEs by making the $\mathcal{O}_{\mathsf{MalUserReg}}$ oracle unavailable to smart card attackers.

At the end, the adversary produces a target epoch ϵ^*, entitlement $\mathsf{ent}^*_{\mathsf{sum}}$, proof π^*_{aud}, and blocklist BL^* (line 4). The game checks if the produced proof is valid (line 5) and determines the maximum explainable entitlement $\mathsf{ent}_{\mathsf{max}}$. The maximum consists of two terms (line 7): (1) the non-revoked honest users for which the adversary called $\mathcal{O}_{\mathsf{Showup}}$ for epoch ϵ^* and blocklist BL^*, and (2) the non-revoked malicious users the adversary controls. The adversary wins if the proof verifies, and the total entitlement exceeds $\mathsf{ent}_{\mathsf{max}}$.

Definition 2. An aid-distribution system is auditable if the following probability is negligible:

$$\mathsf{Succ}^{\mathrm{AUD}}(\mathcal{A}) = \mathsf{Pr}\left[\mathsf{Exp}^{\mathrm{AUD}}_{\mathcal{A}}(1^\ell) = 1\right].$$

Theorem 3. *The smart card system is auditable provided that the digital signature scheme is unforgeable and the discrete logarithm assumption holds in $\mathbb{G}$.*

Proof sketch. By design, each card will output exactly one valid signature per epoch. Because the audit proof verified, only one record per household was included in the proof (otherwise there would be repeated τ_Hs). The adversary cannot forge signatures, so all entries in the proof, including the commitments $\mathsf{Com}_{\mathsf{ent}}$ must be produced by honest tokens. Under the discrete logarithm assumption, Pedersen commitments are computationally binding, so the adversary cannot find another opening to the product of commitments. □

Theorem 4. *The smartphone system is auditable provided that the ABC scheme is unforgeable and the discrete logarithm holds in $\mathbb{G}_1$.*

Algorithm 5 Entitlement privacy

```
1: function $\mathsf{Exp}^{\mathrm{ENT}}_{\mathcal{A},b}(1^\ell)$
2:     $\mathsf{param} \leftarrow \mathsf{GlobalSetup}(1^\ell)$
3:     $\mathsf{sk}, \mathsf{pk}, \leftarrow \mathsf{SetupRS}(1^\ell)$
4:     $\epsilon^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{HonestReg}}(\cdot), \mathcal{O}_{\mathsf{ShowupTwo}}(\cdot)}(\mathsf{pk})$
5:     $\mathsf{ent}^0_{\mathsf{sum}}, \pi^0_{\mathsf{aud}} \leftarrow \mathsf{GenAudPiDS}(\epsilon^*, \mathsf{log}^0, \mathsf{BlockLists}[\epsilon]^*)$
6:     $\mathsf{ent}^1_{\mathsf{sum}}, \pi^1_{\mathsf{aud}} \leftarrow \mathsf{GenAudPiDS}(\epsilon^*, \mathsf{log}^1, \mathsf{BlockLists}[\epsilon]^*)$
7:     if $\mathsf{ent}^0_{\mathsf{sum}} \neq \mathsf{ent}^1_{\mathsf{sum}}$ or $|\pi^0_{\mathsf{aud}}| \neq |\pi^1_{\mathsf{aud}}|$ or
           $|\mathsf{BlockLists}[\epsilon^*]| > 1$ then
8:         return $\perp$
9:     $b' \leftarrow \mathcal{A}(\pi^b_{\mathsf{aud}})$
10:    return $b' = b$
```

Proof sketch. By definition the audit proof verifies. Therefore, all contained proofs π_{ent} must be valid and the corresponding tags must be unique. Since the distribution station cannot forge credentials, it cannot create valid proofs π_{ent} (or rather, the proof π_s inside it) except by interacting with an honest or malicious recipient. Thus, all commitments $\mathsf{Com}_{\mathsf{ent}}$ in the audit proof must correspond to legitimate (honest or malicious) recipients. The proof follows as before. □

6.3. Entitlement Privacy Game

An aid-distribution system should not unnecessarily disclose sensitive information of recipients to auditors (P3$_{\text{auditP}}$). By design, the auditor learns the sum of entitlements and the number of legitimate recipients. We model that the auditor does not learn *more* using a two-world game inspired by *Benaloh's ballot privacy game* [5], see Algorithm 5.

In this game, the registration station and distribution station are honest. As before, the adversary can request the creation of honest recipients using $\mathcal{O}_{\mathsf{HonestReg}}$. The key difference is that the game models *two* (parallel) distribution stations. The adversary can request that honest recipients show up to these stations using the oracle $\mathcal{O}_{\mathsf{ShowupTwo}}$, which takes as input two recipient identifiers, who interact with the respective distribution stations. The $\mathcal{O}_{\mathsf{ShowupTwo}}$ oracle tracks which blocklist the adversary uses in each epoch.

In the entitlement privacy experiment, the adversary outputs a challenge epoch ϵ^* (line 4). The two distribution stations will produce their audit proofs for ϵ^* (lines 5–6). Recall that the auditor always learns the total entitlement, the number of recipients and the blocklist used. Therefore, the adversary loses if these are not the same in the two worlds (lines 7–8). (Note that requiring that BL is the same in each call to $\mathcal{O}_{\mathsf{ShowupTwo}}$ is not sufficient to prevent trivial wins because a call to $\mathsf{ShowupT}$ could fail in only one of the two worlds. This difference would be detectable.) Finally, the auditor receives one of the proofs and needs to guess which of the two it is (lines 9–10). Any extra leakage will let the adversary win the game.

Definition 3. Aid-distribution systems provide entitlement privacy if the following advantage is negligible:

$$\mathsf{Adv}^{\mathrm{ENT}}_{\mathcal{A}} = \left|\mathsf{Pr}\left[\mathsf{Exp}^{\mathrm{ENT}}_{\mathcal{A},0}(1^\ell)\right] - \mathsf{Pr}\left[\mathsf{Exp}^{\mathrm{ENT}}_{\mathcal{A},1}(1^\ell)\right]\right|.$$

Theorem 5. *The systems provide entitlement privacy against a malicious auditor assuming the commitment scheme is hiding (and, for the phone solution, the ABC disclosure proofs are simulatable).*

Proof sketch. For each audit entry, only $\mathsf{Com}_{\text{ent}}$ relates to the recipient's entitlement. This is clear by inspection for the smart card approach. For the smartphone approach, this follows from the privacy of the ABC scheme. Finally, because Pedersen's commitments are information-theoretically hiding, the auditor only learns the total entitlement and the number of entries. It therefore cannot win the game. □

6.4. Security of Aid Distribution

Aid-distribution systems should ensure that illegitimate recipients cannot receive aid ($\text{S2}_{\text{legitimate}}$), and that legitimate recipients cannot receive more aid than entitled (S1_{limit}). The security game, the theorems and proofs are very similar to the auditability proofs (Sect. 6.2), except that the distribution station is honest. We therefore defer them to Appendix C.

7. Performance Evaluation

In this section, we evaluate the suitability of the instantiations in Sections 4 and 5 for real deployment. We focus on the performance of the registration and the distribution phases. In particular, we focus on the computation and the communication operations that happen on the token and may affect user experience. We omit the setup and audit phases, because they can be done offline, and thus, are not a bottleneck. Our code can be found here: https://github.com/spring-epfl/not-yet-another-id-code.

Smart-card-based Solution. We implemented a prototype of the smart-card-based solution on a *NXP J3H145 dual 144k* Java Card [45]. We focus on the functions running on the smart card at registration and distribution, as the station can run on powerful hardware. The measurements include communication cost between card and reader. We report the mean over at least 5 runs. In all cases, the standard error of mean (SEM) is below 1%.

At registration, the card receives $(\mathsf{ent}_H, \mathsf{v}_H)$, of 32 bytes each. Then, it runs $\mathsf{FinishRegT}$ and generates a 32-byte secret k_H. In total, computation and communication at registration runs in under 100 ms.

We determine the cost of running $\mathsf{ShowupT}$ by measuring the cost of the individual operations and the transfer cost. First, the card downloads the period ϵ (8 bytes) and the blocklist BL. Assuming 512 entries of 32 bytes, transferring, checking, and hashing the blocklist take 3.3 s. The time scales linearly in the length of the blocklist. Second, the card updates the counter ϵ_{last} and computes a 32-byte tag τ_H. We implement the PRF for computing τ_H using AES in ECB mode. This step takes 113 ms. Third, the card computes the Pedersen commitment on the entitlement. Since the NXP J3H145 dual 144k card does not expose a direct elliptic-curve API, we use JCMathLib [42] and the card's Diffie-Hellman key exchange functionality to compute the commitment. Computing $\mathsf{Com}_{\text{ent}}$ takes 500 ms. Finally, signing takes 166 ms. Adding these together (except the first one), we estimate the baseline cost of $\mathsf{ShowupT}$ to be around 779 ms.

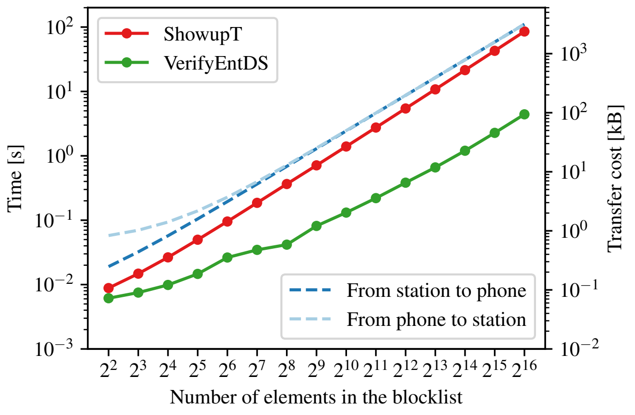

Figure 3. Smartphone solution: Performance at distribution. The computation cost of $\mathsf{ShowupT}$ (red), computation cost of $\mathsf{VerifyEntDS}$ (green), the transfer cost of station→phone (blue), and the transfer cost of phone→station (light blue). All are linear in the length of the blocklist.

With 512 entries on the blocklist, the distribution protocol takes less than 5 seconds in total. This running time is smaller than any physical interaction happening at distribution time, and hence our protocol can fulfill D2_{scale}.

Smartphone-based Solution. We implemented a Rust [51] prototype of the smartphone solution. Our prototype includes the Pedersen Commitment scheme [46] and the Pointcheval-Sanders scheme [47] using SHA-256 as the hash function. In our experiments we use a *Samsung Galaxy a40* smartphone with a *Samsung Exynos 7 Octa 7904* chipset for recipients, and a computer with an *Intel(R) Core(TM) i7-7600U CPU 2.80GHz* as the stations. We report running times (averaged over 16 runs) and transfer costs. Transfer times will depend on the communication channel. For example, Bluetooth can achieve 1 Mb/s. Appendix A discusses other channels.

At registration the phone runs $\mathsf{PrepareRegT}$ and $\mathsf{FinishRegT}$ which take 4.6 ms and 0.6 ms respectively. The registration station runs $\mathsf{ProcessRegRS}$ which takes 0.8 ms. The request sent from the phone to the station and the response from the station to the phone are both less than a few hundred bytes.

Figure 3 shows the computation cost of running $\mathsf{ShowupT}$ on the phone, the computation cost of verifying the proof ($\mathsf{VerifyEntDS}$) on the station, as well as the communication costs. Even with 1024 blocked households, the whole distribution protocol can still finish within seconds, making this solution efficient enough for supporting aid distribution at scale (D2_{scale}).

8. Practical Considerations

Multiple Registration and Distribution Stations. While our design focuses on cases where there is only one registration and distribution station, humanitarian organizations may need to set up multiple registration and distribution sites for large-scale operations [27]. For example, having multiple sites reduces the waiting time and therefore lowers the risk of attack for staff and recipients [27]. But having multiple sites can also affect the security and privacy offered of the system.

When there are multiple registration stations, recipients from the same household can register several times and get goods multiple times per period. To avoid this problem, registration stations need access to the validation means of other registration stations (e.g., communication with the elders of another village).

When there are multiple distribution sites, recipients can ask for goods in more than one of them. To avoid this problem, distribution stations need to synchronize their lists of seen tags τ_H to learn which households already requested goods in a period. Synchronization requires communication, e.g., Internet connectivity or transportation of digital copies (e.g., in a USB stick) from one site to the other. Communication may not be possible, or be difficult in some emergency scenarios where aid distribution takes place.

In absence of connectivity, assigning each household to a fixed station would prevent multiple-station-based double dipping at the cost of flexibility for recipients.

Preventing Delegation of Tokens. One of the problems that the ICRC staff finds in the field is that recipients may willingly or unwillingly, give their entitlement proofs to others. When this happens, illegitimate recipients gain access to goods. To avoid this issue, the distribution station needs to authenticate recipients to ensure that the recipient is indeed registered at the registration station.

The most popular authentication methods are not suitable in our case. For example, (graphical) passwords or tokens can be delegated and thus do not solve the problem. Due to the ease of use and the inherent protection against delegation, biometrics are perceived by the humanitarian sector as a very advantageous solution.

Biometric-based authentication, while desirable, also raises concerns. Biometrics contain private information about individuals, and they are not renewable. Once they are leaked or shared, those having the biometric data can use them to re-identify their owners forever. Humanitarian organizations are already under pressure from state and non-state actors to share or disclose biometric data they have [23]. This is why, to reduce risks for recipients, the ICRC has a strict policy for the usage of biometrics [31] which recommends avoiding the creation of biometrics databases. Such biometrics policy is in line with other relevant legal frameworks under which humanitarian organizations may have to operate, e.g., the EU General Data Protection Regulation [17] or the African Union Convention on Cybersecurity and Personal Data Protection [1].

Our design can be easily adapted to support biometric authentication while respecting the ICRC policies. It suffices to store the biometrics on the tokens at registration and implementing the biometric authentication *inside* the token. At the distribution station, the user can use the token to prove their identity. To keep the system properties, it is necessary that the biometric sensors and devices realizing the computation (smart cards or smartphones) are trusted.

Limits of technological solutions against delegation. While biometrics (and other strong authentication mechanisms) can prevent delegation, they cannot fully prevent illegitimate recipients from getting access to aid. For example, recipients may be coerced to give out their aid. This problem is hard to solve with technology, as digital systems can check the authenticity or eligibility of the recipient, but not their willingness.

9. Conclusion

Humanitarian organizations see digitalization as an opportunity to increase their operations' efficiency and therefore increase their capability to help people in need. However, they also recognize that the introduction of technology may bring new risks for the populations they serve [36].

In this paper we tackled the case of digitalization of aid-distribution programs. In collaboration with the ICRC we have identified the requirements that a digital aid-distribution solution must fulfill to guarantee that it preserves the safety, rights, and dignity of aid recipients. Then, we propose a decentralized aid-distribution system that enables humanitarian organizations to distribute physical goods at scale in a secure and privacy-preserving manner, while providing strong accountability.

We provide two instantiations of our design on two kinds of digital tokens already used in humanitarian contexts: smart cards and smartphones. We formally prove that our schemes provide the required security and privacy properties, and we empirically demonstrate that, despite the use of advanced cryptography, they are efficient enough to support mass aid-distribution.

Our interactions with the ICRC reveal that, due to them dealing with the most vulnerable populations, the needs of humanitarian organizations often cannot be satisfied with off-the-shelf commercial solutions. Their requirements often bring up challenging research questions and open new design spaces rarely explored by our community. We hope that our work fosters new collaborations between researchers and humanitarian organizations to explore this space so research innovations can directly benefit those most in need of help.

Acknowledgements

We thank Laurent Girod for implementing the Java Card prototype and Nathan Duchesne for implementing the smartphone prototype.

This work was partially funded by the Science and Technology for Humanitarian Action Challenges (HAC) programme from the Engineering for Humanitarian Action initiative, a partnership between the ICRC, EPFL and ETHZ.

References

[1] African Union, "African Union Convention on Cyber Security and Personal Data Protection," https://au.int/en/treaties/african-union-convention-cyber-security-and-personal-data-protection, 2014, accessed: April 8, 2023.

[2] M. H. Au, W. Susilo, Y. Mu, and S. S. M. Chow, "Constant-size dynamic k-times anonymous authentication," *IEEE Syst. J.*, 2013.

[3] F. Baldimtsi and A. Lysyanskaya, "Anonymous credentials light," in *CCS*, 2013, pp. 1087–1098.

[4] A. Barua, M. A. A. Alamin, M. S. Hossain, and E. Hossain, "Security and Privacy Threats for Bluetooth Low Energy in IoT and Wearable Devices: A Comprehensive Survey," *IEEE Open J. Commun. Soc.*, 2022.

[5] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi, "SoK: A Comprehensive Analysis of Game-Based Ballot Privacy Definitions," in *S&P*, 2015.

[6] S. L. Blond, A. Cuevas, J. R. Troncoso-Pastoriza, P. Jovanovic, B. Ford, and J. Hubaux, "On Enforcing the Digital Immunity of a Large Humanitarian Organization," in *S&P*, 2018.

[7] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *CCS*, 2004.

[8] S. A. Brands, *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, 2000.

[9] E. F. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *CCS*, 2004.

[10] J. Burton, ""Doing no harm" in the digital age: What the digitalization of cash means for humanitarian action," https://international-review.icrc.org/sites/default/files/reviews-pdf/2021-03/doing-no-harm-digitalization-of-cash-humanitarian-action-913.pdf, 2020, accessed: April 8, 2023.

[11] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *SCN*, 2002.

[12] M. Cäsar, T. Pawelke, J. Steffan, and G. Terhorst, "A survey on Bluetooth Low Energy security and privacy," *Comput. Networks*, 2022.

[13] M. Chase, S. Meiklejohn, and G. Zaverucha, "Algebraic MACs and Keyed-Verification Anonymous Credentials," in *CCS*, 2014.

[14] M.-L. Clausen and B. O. Martins, "Humanitarian Biometrics in Yemen: The complex politics of humanitarian technology," https://blogs.prio.org/2021/07/humanitarian-biometrics-in-yemen-the-complex-politics-of{-}humanitarian-technology/, 2021, accessed: April 8, 2023.

[15] I. Damgård and N. Triandopoulos, "Supporting Non-membership Proofs with Bilinear-map Accumulators," *IACR Cryptol. ePrint Arch.*, 2008.

[16] A. Davidson, I. Goldberg, N. Sullivan, G. Tankersley, and F. Valsorda, "Privacy Pass: Bypassing Internet Challenges Anonymously," *PoPETS*, 2018.

[17] European Union, "General Data Protection Regulation (EU) (GDPR)," https://eur-lex.europa.eu/eli/reg/2016/679/oj, 2016, accessed: April 8, 2023.

[18] E. Fenske, D. Brown, J. Martin, T. Mayberry, P. Ryan, and E. C. Rye, "Three Years Later: A Study of MAC Address Randomization In Mobile Devices And When It Succeeds," *PoPETS*, 2021.

[19] FIDO Alliance, "FIDO 2.0: Overview," https://fidoalliance.org/specs/fido-v2.0-rd-20170927/fido-overview-v2.0-rd-20170927.html, 2017.

[20] J. Fussell, "Group Classification on National ID Cards as a Factor in Genocide and Ethnic Cleansing," http://www.preventgenocide.org/prevent/removing-facilitating-factors/IDcards/, 2001, accessed: April 8, 2023.

[21] D. Gao, H. Lin, Z. Li, F. Qian, Q. A. Chen, Z. Qian, W. Liu, L. Gong, and Y. Liu, "A nationwide census on wifi security threats: prevalence, riskiness, and the economics," in *ACM MobiCom*, 2021.

[22] D. Gentry and A. Pennarun, "Passive Taxonomy of Wifi Clients Using MLME Frame Contents," *CoRR*, 2016.

[23] B. Hayes and M. Marelli, "Facilitating innovation, ensuring protection: the ICRC Biometrics Policy," https://blogs.icrc.org/law-and-policy/2019/10/18/innovation-protection-icrc-biometrics-policy/, 2019, accessed: April 8, 2023.

[24] R. Henry and I. Goldberg, "Thinking inside the BLAC box: smarter protocols for faster anonymous blacklisting," in *WPES*, 2013.

[25] G. Hosein and C. Nyst, "Aiding Surveillance," Privacy International, Tech. Rep., 20013.

[26] Human Rights Watch, "New Evidence that Biometric Data Systems Imperil Afghans," https://www.hrw.org/news/2022/03/30/new-evidence-biometric-data-systems-imperil-afghans, 2022, accessed: April 8, 2023.

[27] ICRC, "EcoSec Response," https://shop.icrc.org/ecosec-response-en-pdf.html, accessed: April 8, 2023.

[28] ——, "Targeting, Selection, and Prioritization Methods for Economic Security Programmes," https://shop.icrc.org/download/ebook?sku=4567/002-ebook, accessed: April 8, 2023.

[29] ——, "Guidelines for Cash Transfer Programming," https://www.icrc.org/en/doc/assets/files/other/icrc_002_mouvement-guidelines.pdf, 2007, accessed: April 8, 2023.

[30] ——, "The ICRC: Its Mission and Work," https://shop.icrc.org/download/ebook?sku=0963/002-ebook, 2009, accessed: April 8, 2023.

[31] ——, "Policy on the Processing of Biometric Data by the ICRC," https://www.icrc.org/en/download/file/106620/icrc_biometrics_policy_adopted_29_august_2019_.pdf, 2019, accessed: April 8, 2023.

[32] ——, "Evidence-based Decision-making in Delegations: Analysis and Evidence Planning Guidance," https://www.icrc.org/en/publication/4525-strengthening-evidence-based-decision-making-delegations\protect\penalty\z@-analysis-and-evidence, 2021, accessed: April 8, 2023.

[33] ——, "ICRC Annual Report Africa 2021," https://www.icrc.org/en/download/file/247666/icrc-annual-report-africa_2021_forweb.pdf, 2021, accessed: April 8, 2023.

[34] ——, "The ICRC's funding and spending," https://www.icrc.org/en/faq/icrcs-funding-and-spending, 2022, accessed: April 8, 2023.

[35] ISO/IEC 18004:2015, "Information technology — Automatic identification and data capture techniques — QR Code bar code symbology specification," 2015.

[36] A. Kaspersen and C. Lindsey-Curtet, "The digital transformation of the humanitarian sector," https://blogs.icrc.org/law-and-policy/2016/12/05/digital-transformation-humanitarian-sector/, 2016, accessed: April 8, 2023.

[37] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[38] K. Krombholz, P. Frühwirt, P. Kieseberg, I. Kapsalis, M. Huber, and E. R. Weippl, "QR Code Security: A Survey of Attacks and Challenges for Usable Security," in *HAS*, 2014.

[39] J. Lapon, M. Kohlweiss, B. D. Decker, and V. Naessens, "Analysis of revocation strategies for anonymous idemix credentials," in *Communications and Multimedia Security*, 2011.

[40] J. Li, N. Li, and R. Xue, "Universal Accumulators with Efficient Nonmembership Proofs," in *ACNS*, 2007.

[41] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown, "A Study of MAC Address Randomization in Mobile Devices and When it Fails," *PoPETS*, 2017.

[42] V. Mavroudis and P. Svenda, "JCMathLib: Wrapper Cryptographic Library for Transparent and Certifiable JavaCard Applets," in *SPW*, 2020.

[43] NADRA, "National Database and Registration Authority - ESCAP," https://www.unescap.org/sites/default/d8files/event-documents/National_Database_Registration_Authority_Inception_ws_Pakistan_11Feb2022.pdf, 2022, accessed: April 8, 2023.

[44] L. Nguyen, "Accumulators from Bilinear Pairings and Applications," in *CT-RSA*, 2005.

[45] Oracle, "Java Card Development Kit User Guide," https://docs.oracle.com/en/java/javacard/3.1/guide/, 2021, accessed: April 8, 2023.

[46] T. P. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," in *CRYPTO*, 1991.

[47] D. Pointcheval and O. Sanders, "Short Randomizable Signatures," in *CT-RSA*, 2016.

[48] D. Rupprecht, A. Dabrowski, T. Holz, E. R. Weippl, and C. Pöpper, "On Security Research Towards Future Mobile Network Generations," *IEEE Commun. Surv. Tutorials*, 2018.

[49] T. Schwartz, "A Model for Humanitarian Aid Beneficiary Targeting," https://timothyschwartzhaiti.com/a-model-for-humanitarian-aid-beneficiary-targeting/, 2019, accessed: April 8, 2023.

[50] L. Taylor and D. Broeders, "In the Name of Development: Power, Profit and the Datafication of the Global South," *Geoforum*, 2015.

[51] The Rust Foundation, "Rust," https://github.com/rust-lang/rust, 2022, accessed: April 8, 2023.

[52] The World Bank Group, "Beneficiary Assessment," http://web.worldbank.org/archive/website00519/WEB/OTHER/BENEFICI.HTM, 2004, accessed: April 8, 2023.

[53] Trilateral Research & Consulting, "Privacy Impact Assessment of UNHCR Cash Based Interventions," https://www.calpnetwork.org/wp-content/uploads/2020/01/privacy-impact-assessment-of-unhcr-cash-based-interventions.pdf, 2015, accessed: April 8, 2023.

[54] P. P. Tsang, M. H. Au, A. Kapadia, and S. W. Smith, "BLAC: Revoking Repeatedly Misbehaving Anonymous Users without Relying on TTPs," *ACM Trans. Inf. Syst. Secur.*, 2010.

[55] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms," in *AsiaCCS*, 2016.

[56] H. A. M. Wahsheh and F. L. Luccio, "Security and Privacy of QR Code Applications: A Comprehensive Study, General Guidelines and Solutions," *Inf.*, 2020.

[57] B. Wang, W. Lueks, J. Sukaitis, V. Graf Narbel, and C. Troncoso, "Not Yet Another Digital ID: Privacy-preserving Humanitarian Aid Distribution," *CoRR*, vol. abs/2303.17343, 2023.

[58] World Food Programme, UNHCR, "Joint Inspection of the Biometrics Identification System for Food Distribution in Kenya," https://documents.wfp.org/stellent/groups/public/documents/reports/wfp277842.pdf, 2015, accessed: April 8, 2023.

[59] L. Yu, B. Luo, J. Ma, Z. Zhou, and Q. Liu, "You Are What You Broadcast: Identification of Mobile and IoT Devices from (Public) WiFi," in *USENIX Security*, 2020.

Appendix A. Privacy-friendly Smartphone-Station Channel

When using a smart card as a token, the communication between the token and the stations happens through a card reader. Such a communication channel is hard to eavesdrop, and as long as the card reader is honest, hard to compromise. When using a smartphone as a token, the communication becomes wireless. Wireless communications are is easy to eavesdrop on and, depending on the protocol, are also susceptible to person-in-the-middle attacks. This new attack surface can have a strong impact on privacy.

We discuss four widely used communication channels suitable for connecting the smartphone to the station when deploying the system: cellular network, local Wi-Fi, Bluetooth, and QR-code scanning. We compare them in terms of (1) ease of integration into a humanitarian aid-distribution system and (2) the privacy risks they introduce.

Cellular Network. Smartphones can use cellular networks to connect to the Internet. If the stations also have Internet connection, the phone can communicate with the station via this channel. Using cellular networks has the advantage that it does not require the installation of dedicated hardware to support the communication. However, one cannot always assume that the areas where humanitarian aid distribution takes place have reliable and high-bandwidth cellular connection system.

The use of cellular networks extends the threat model of the system to include the mobile service provider, as well as eavesdroppers with adequate equipment [48]. These adversaries can link recipients, and hence, breach their privacy.

Local Wi-Fi. An alternative for connecting smartphones and stations is to use a local Wi-Fi network to which both devices connect. This method requires the deployment and maintenance of Wi-Fi routers. Once these routers are deployed, quality of connection is easy to achieve.

Using a Wi-Fi eliminates the need to trust a mobile service provider. Instead, trust is put on the entity setting up the router to not track users [21], [59]. If this entity is not trustworthy, users need to have unlinkable MAC addresses, which are not available in all commercial devices [41], and prevent fingerprinting of other information contained in the probe request [18], [22], [55].

Bluetooth. A third option is to use the Bluetooth Low Energy (BLE) technology [4], [12]. It enables smartphones to connect to the station without intermediaries that could be adversarial. Compared to Wi-Fi, BLE provides less bandwidth, but is still enough to run our protocols. On the negative side, BLE does not have the same prevalence on phones as Wi-Fi.

With regard to privacy, the BLE specification contains provisions to randomize the MAC address of the devices. However, enabling full randomization to ensure anonymity from the receiving end comes at the cost of having to pair the devices every time. Whether frequent re-pairing is acceptable depends on the periodicity of the distribution.

QR-code Scanning. QR codes are two-dimensional bar codes containing data [35]. Users can access the encoded data by scanning the QR code with the camera of their smartphones [38]. Due to their ease of deployment, QR codes are increasingly popular in smartphone settings [56], e.g., to provide links to social media accounts, to show vaccination certificates, or to share files.

Because it requires the adversary to have direct line of sight to the QR code to capture the encoded data [56], QR-

TABLE 1. Comparing Smartphone-Station Channel

	Cellular Network	Local Wi-Fi	BLE	QR-code
Privacy	low	medium	medium	high
Throughput	high	high	medium	low
Infrastructure	✗	✓	✗	✓
User-friendliness	✓	✓	✓	✗

code scanning provides a high level of privacy. However, the storage capacity of a QR code is limited (a maximum of 3 kB [35]). Thus, more than one QR code may be needed to transmit all data necessary in our protocols. Even though we can rotate QR codes to transfer more data, the throughout is still low due to other constraints, e.g, the recognition delay of the phone, people unlocking the phone to point the camera to the code, etc. This may slow down the protocol's execution rendering this method impractical in reality (D2scale).

Table 1 summarizes the performance of the four channels. If the blocklist is short and only requires low throughput of the channel, QR-code scanning can be a good option because of better privacy. If the blocklist is longer and requires larger throughput, BLE or local Wi-Fi would be a more desirable solution, depending on the ability of setting up Wi-Fi device. BLE has an advantage over Wi-Fi if devices can enable fully randomization of MAC addresses. Cellular network should be the last alternative due to the large privacy risk. In the case where the privacy risk is unacceptable, one option is to move to smart-card-based solution, another option is to shorten the distribution period to encourage having a shorter blocklist.

Appendix B. Cryptography Details

B.1. Digital Signatures

Let $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. We restate the existential unforgeability game $\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(\ell)$ for adversary $\mathcal{A}$ and security parameter ℓ:

1) The challenger runs $\mathsf{Gen}(1^\ell)$ to obtain a signing-verification key-pair $(\mathsf{sk}, \mathsf{pk})$.
2) The adversary $\mathcal{A}$ is given pk and has access to the signing oracle $\mathsf{Sign}(\mathsf{sk}, \cdot)$.
3) Eventually $\mathcal{A}$ outputs a forgery (σ, m). Let $\mathcal{Q}$ denote the set of all message queries that $\mathcal{A}$ asked its oracle. The adversary succeeds if and only if (1) $\mathsf{Verify}(\mathsf{pk}, \sigma, m) = \top$ and (2) $m \notin \mathcal{Q}$.
4) The experiment outputs 1 if the adversary wins.

A signature scheme $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ is *unforgeable* if for all probabilistic polynomial-time adversaries $\mathcal{A}$, there is a negligible function negl such that:

$$\mathsf{Pr}\big[\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n) = 1\big] \leqslant \mathsf{negl}(n).$$

B.2. Pseudorandom Functions

Pseudorandom functions (PRFs) are "random-looking functions" which refer to the pseudorandomness of a distribution on functions [37]. The set Func_n are all functions mapping n-bit strings to n-bit strings.

Definition 4. An efficient, length-preserving, keyed function $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a pseudorandom function if no probabilistic polynomial-time distinguisher D can differentiate F from f such that $f \in \mathsf{Func}_n$, i.e.,

$$|\mathsf{Pr}[D^{F_k(\cdot)}(1^n) = 1] - \mathsf{Pr}[D^{f(\cdot)}(1^n) = 1]| \leqslant \mathsf{negl}(n),$$

where the first probability is taken over uniform choice of $k \in_R \{0,1\}^n$ and the randomness of D.

B.3. Pointcheval-Sanders Credentials

There are three parties in an ABC scheme: *the issuer*, *the user*, and *the verifier*. The issuer sets up the system and issues credentials to users. The process by which a user obtains a credential is called *issuance*. The user holds credentials and shows them to the verifier. The user can choose to reveal some attributes from any number of the credentials to the verifier. The verifier checks the credential is valid, and the revealed attributes fulfill the requirements of the application. The process by which the user shows possession of a credential to the verifier is called *verification*.

A secure ABC scheme has the following properties:

- Unforgeability: It is not possible for any party in the system to forge a credential without the help of the issuer.
- Unlinkability: It is not possible for the verifier (even when colluding with the issuer) to distinguish between two users who disclose the same attributes in the verification process.

B.3.1. Setup. The issuer runs $\mathsf{ABC.Gen}$ to set up global parameters and generate keys. It proceeds as follows:

1) The algorithm takes as input the security parameter 1^ℓ and the number of attributes L.
2) It generates public parameters containing a type-III bilinear group pair given by $\mathsf{param}_{\mathsf{PS}} = (e(\cdot,\cdot), \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, g_1, g_2, g_T)$ where g_1, g_2, g_T are generators of the groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order q respectively.
3) It generate signing and verification keys. It picks $x, y_1, \ldots, y_L \in_R \mathbb{Z}_q$, $g \in_R \mathbb{G}_1$ and $\tilde{g} \in_R \mathbb{G}_2$, then computes:

$$X = g^x, \tilde{X} = \tilde{g}^x, Y_i = g^{y_i}, \tilde{Y}_i = \tilde{g}^{y_i}$$

for $i = 1, \ldots, L$. It publishes the public key $\mathsf{pk}_{\mathsf{PS}}$ and returns the private key $\mathsf{sk}_{\mathsf{PS}}$, where

$$\mathsf{pk}_{\mathsf{PS}} = (g, Y_1, \ldots, Y_L, \tilde{g}, \tilde{X}, \tilde{Y}_1, \ldots, \tilde{Y}_L)$$
$$\mathsf{sk}_{\mathsf{PS}} = (x, X, y_1, \ldots, y_L).$$

B.3.2. Issuance. The user and the issuer jointly run $\mathsf{ABC.issue}$ protocol to create signatures as credentials. The issuer acts as a signer without knowing all the attributes it is signing. The protocol proceeds as follows:

1) *Input agreement.* The user and the issuer agree on the set of attribute indices $\mathcal{I} \subset \{1, \dots, L\}$ that are determined by the issuer and the set of attribute indices $\mathcal{U} \subset \{1, \dots, L\}$ that are determined by the user, where $\mathcal{I} \cup \mathcal{U} = \{1, \dots, L\}$. The user takes as input the public key $\mathsf{pk}_{\mathsf{PS}}$ and the attributes $a_i, \forall i \in \mathcal{U}$. The issuer takes as input the public key $\mathsf{pk}_{\mathsf{PS}}$, the private key $\mathsf{sk}_{\mathsf{PS}}$, and the attributes $a_i, \forall i \in \mathcal{I}$.
2) *User commitment.* The user commits to the attributes they want to include in the credential by picking $t \in_R \mathbb{Z}_q$ at random and computing the credential C as well as a non-interactive zero-knowledge proof (NIZK) π that proves C has been computed correctly:
$$C = g^t \prod_{i \in \mathcal{U}} Y_i^{a_i},$$
$$\pi = \mathsf{NIZK}\left\{(t, (a_i)_{i \in \mathcal{U}}) : C = g^t \prod_{i \in \mathcal{U}} Y_i^{a_i}\right\}.$$
The user sends an issuance request $\mathsf{request}_{\mathsf{abc}} = (C, \pi)$ to the issuer.
3) *Issuer signing.* The issuer receives the issuance request $\mathsf{request}_{\mathsf{abc}} = (C, \pi)$, verifies the validity of proof π with respect to commitment C, and aborts if the proof is not correct. Otherwise, the issuer picks $u \in_R \mathbb{Z}_q$ at random and creates the signature
$$\sigma' = \left(g^u, \left(XC \prod_{i \in \mathcal{I}} Y_i^{a_i}\right)^u\right).$$
The issuer sends σ' and the attributes chosen by the issuer $a_i, \forall i \in \mathcal{I}$ to the user.
4) *Unblinding signature.* The user receives the signature $\sigma' = (\sigma'_1, \sigma'_2)$ and attributes. The user computes the "unblinding" signature
$$\sigma = \left(\sigma'_1, \frac{\sigma'_2}{(\sigma'_1)^t}\right),$$
and uses the public key pk to validate that σ is a valid signature on the attributes. If valid, the user stores both σ and the attributes.

When using the ABC scheme in our protocols for the smartphone solution, the phone additionally proves that the revocation value $\mathsf{v}_H = (g_1, g_1^{r_H}) = (g_1, \overline{\mathsf{v}_H})$ has been correctly formed. The issuer can trivially check that the first component equals the generator of $\mathbb{G}_1$. The token extends the proof of correctness π above to show that the second component is correct. Instantiating the user-defined attributes at positions 1 and 3, it instead proves:
$$\pi = \mathsf{NIZK}\left\{(t, \mathsf{k}_H, r_H) : C = g^t Y_1^{\mathsf{k}_H} Y_3^{r_H} \wedge \overline{\mathsf{v}_H} = g_1^{r_H}\right\}$$

B.3.3. Verification. The user can run $\mathsf{ABC.show}$ to convince the verifier that the user possesses a valid credential over a set of attributes using a zero-knowledge proof. As part of this proof, the user can choose to reveal some of the attributes while hiding others from the verifier. The verifier checks and accepts the proof if all the checks succeed. The showing protocol proceeds as follows:

1) *Input agreement.* The user and the verifier agree on the public key of the issuer and the set of attributes $\mathcal{D}$ that should be disclosed to the verifier. Let $\mathcal{H} = \{1, \dots, L\} \setminus \mathcal{D}$ be the set of attributes that are hidden from the verifier.
2) *Proof creation.* The user takes as input a signature $\sigma = (\sigma_1, \sigma_2)$ over the attributes $a_1, \dots, a_L$ and picks random values $r_s, t_s \in_R \mathbb{Z}_q$. The user computes a randomized signature σ_s and a non-interactive zero-knowledge proof that proves the signature σ_s is a valid signature:
$$\sigma_s = (\sigma_{s_1}, \sigma_{s_2}) = (\sigma_1^{r_s}, (\sigma_2 \sigma_1^{t_s})^{r_s}),$$
$$\pi_s = \mathsf{NIZK}\{(t_s, (a_i)_{i \in \mathcal{H}}) : \frac{e(\sigma_{s_2}, \tilde{g}) \prod_{i \in \mathcal{D}} e(\sigma_{s_1}, \tilde{Y}_i)^{-a_i}}{e(\sigma_{s_1}, \tilde{X})} = e(\sigma_{s_1}, \tilde{g})^{t_s} \prod_{i \in \mathcal{H}} e(\sigma_{s_1}, \tilde{Y}_i)^{a_i}.\}$$
The user sends $(\sigma_s, (a_i)_{i \in \mathcal{D}}, \pi_s)$ to the verifier.
3) *Proof verification.* The verifier receives the signature $\sigma_s = (\sigma_{s_1}, \sigma_{s_2})$, the disclosed attributes $(a_i)_{i \in \mathcal{D}}$, and the proof π_s from the user. First, the verifier checks that $\sigma_{s_1} \neq 1_{\mathbb{G}_1}$, i.e., σ_{s_1} is not the unity element in $\mathbb{G}_1$. Second, the verifier checks the user has a valid signature under the public key $\mathsf{pk}_{\mathsf{PS}}$ over the disclosed attributes $(a_i)_{i \in \mathbb{D}}$ by verifying the proof π_s. The verifier accepts the disclosure proof only if both checks pass.

Appendix C. Security of Aid Distribution

We model security in the sense of S1$_{\text{limit}}$ and S2$_{\text{legitimate}}$ in a very similar manner to the auditabilty game, but with more restrictions on the adversary's power. In the security game, the adversary controls only users, and must convince an *honest* distribution station to hand out more aid than permitted. Because the adversary no longer controls the distribution station, we give it access to an explicit $\mathcal{O}_{\mathsf{VerifyEnt}}$ oracle in the security game in Algorithm 6 through which it can simulate users (honest or malicious) showing up at the distribution station.

The adversary can, as before, create malicious users (using $\mathcal{O}_{\mathsf{MalUserReg}}$) as well as honest users (using $\mathcal{O}_{\mathsf{HonestReg}}$). It can use $\mathcal{O}_{\mathsf{Showup}}$ to obtain show up data of honest users, which it could then feed into $\mathcal{O}_{\mathsf{VerifyEnt}}$. See line 7. We model the fact that smart cards are TEEs, by making the $\mathcal{O}_{\mathsf{MalUserReg}}$ and $\mathcal{O}_{\mathsf{Showup}}$ oracles unavailable to smart card attackers.

The oracle $\mathcal{O}_{\mathsf{VerifyEnt}}$ tracks show-up events that the distribution station would accept in log. The adversary's goal is to convince the distribution station to, for a given round and

Algorithm 6 Security experiment

```
1: function $\mathcal{O}_{\textsf{VerifyEnt}}(\epsilon, \mathsf{ent}_H, \tau_H, \pi_{\text{ent}}, \mathsf{BL})$
2:     if $\mathsf{VerifyEntDS}(\mathsf{pk}, \epsilon, (\mathsf{ent}_H, \tau_H, \pi_{\text{ent}}), \mathsf{BL}) \wedge \tau_H \notin \mathsf{log}$ then
3:         $\mathsf{log}[\epsilon] \leftarrow \mathsf{log}[\epsilon] \cup \{(\mathsf{ent}_H, \tau_H, \pi_{\text{ent}}, \mathsf{BL})\}$
4: function $\mathsf{Exp}^{\text{SEC}}_{\mathcal{A}}(1^\ell)$
5:     $\mathsf{param} \leftarrow \mathsf{GlobalSetup}(1^\ell)$
6:     $\mathsf{sk}, \mathsf{pk}, \leftarrow \mathsf{SetupRS}(1^\ell)$
7:     $\epsilon^*, \mathsf{BL}^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{HonestReg}}(\cdot), \mathcal{O}_{\mathsf{MalUserReg}}(\cdot), \mathcal{O}_{\mathsf{Showup}}(\cdot), \mathcal{O}_{\mathsf{VerifyEnt}}(\cdot)}(\mathsf{pk})$
8:     $\mathsf{ent}_{\text{seen}} \leftarrow \sum\{\mathsf{ent}_H \mid (\mathsf{ent}_H, \cdot, \cdot, \mathsf{BL}^*) \in \mathsf{log}[\epsilon^*]\}$
9:     $\mathcal{R}_{\text{mal}} = \{\mathsf{id} \mid \mathsf{Rev}[\mathsf{id}] \in \mathsf{BL}^* \wedge \mathsf{id} \in \mathcal{M}\}$
10:    $\mathsf{ent}_{\text{max}} \leftarrow \mathsf{ent}_{\text{sum}}[\epsilon^*, \mathsf{BL}^*] + \sum_{\mathsf{id} \in \mathcal{M} \setminus \mathcal{R}_{\text{mal}}} \mathsf{Ent}[\mathsf{id}]$
11:    return $\mathsf{ent}_{\text{seen}} > \mathsf{ent}_{\text{max}}$
```

blocklist, to hand out more aid than the joint entitlement of all non-revoked malicious users controlled by the adversary *and* non-revoked honest users that showed up. This model then covers: illegitimate recipients receiving aid; legitimate recipients receiving more aid than entitled; a group of colluding users receiving more aid than they are jointly entitled to; and revoked users receiving aid.

After interacting with oracles, the adversary outputs a target epoch ϵ^* and a blocklist BL^*. First, the challenger computes the total amount of aid handed out in terms of successful calls to $\mathcal{O}_{\mathsf{VerifyEnt}}$ for this choice of blocklist (line 8). Then, as in the indistinguishability game, we compute the maximum allowable entitlement $\mathsf{ent}_{\text{max}}$ in two parts (see line 10). The $\mathsf{ent}_{\text{sum}}$ term captures the aid requested by non-revoked honest users using $\mathcal{O}_{\mathsf{Showup}}$. The second term accounts for the non-revoked malicious users (see line 9). If the accepted aid is larger than $\mathsf{ent}_{\text{max}}$, the adversary wins.

Definition 5. An aid-distribution system is secure if the following probability is negligible:

$$\mathsf{Succ}^{\text{SEC}}(\mathcal{A}) = \Pr\left[\mathsf{Exp}^{\text{SEC}}_{\mathcal{A}}(1^\ell) = 1\right].$$

We prove the security of distribution in both solutions using a reduction to auditability.

Theorem 6. *If a scheme is auditable, this scheme also provides security of distribution.*

Proof. Assuming there exists an adversary $\mathcal{A}$ that breaks the security of distribution, i.e., $\mathcal{A}$ can win $\mathsf{Exp}^{\text{SEC}}_{\mathcal{A}}$ in Algorithm 6 with non-negligible advantage, we construct an adversary $\mathcal{B}$ that breaks auditability by winning $\mathsf{Exp}^{\text{AUD}}_{\mathcal{A}}$ in Algorithm 4 with at least the same advantage.

At the start of the auditability game, $\mathcal{B}$ receives param and the public key pk. It relays these to the security adversary $\mathcal{A}$. Whenever $\mathcal{A}$ calls the three oracles, i.e., $\mathcal{O}_{\mathsf{HonestReg}}$, $\mathcal{O}_{\mathsf{MalUserReg}}$, and $\mathcal{O}_{\mathsf{Showup}}$, $\mathcal{B}$ will relay the query to its challenger and send the response back to $\mathcal{A}$. Whenever $\mathcal{B}$ receives a call to $\mathcal{O}_{\mathsf{VerifyEnt}}$ from $\mathcal{A}$, it executes this oracle as stated, and stores the result in $\mathsf{log}[\epsilon]$.

After the oracle queries, $\mathcal{A}$ outputs a chosen period ϵ^* and a target blocklist BL^*. Auditability adversary $\mathcal{B}$ uses all recorded responses in $\mathsf{log}[\epsilon^*]$ to compute $\mathsf{ent}^*_{\text{sum}}, \pi^*_{\text{aud}} \leftarrow \mathsf{GenAudPiDS}(\epsilon, \mathsf{log}[\epsilon^*])$ and finally outputs $(\epsilon^*, \mathsf{ent}^*_{\text{sum}}, \pi^*_{\text{aud}}, \mathsf{BL}^*)$. Assuming this audit proof is valid, $\mathcal{B}$ wins the audit game because $\mathsf{ent}_{\text{seen}} > \mathsf{ent}_{\text{max}}$, and hence, $\mathsf{ent}^*_{\text{sum}} > \mathsf{ent}_{\text{max}}$.

Finally, we show that π^*_{aud} is valid. The verification oracle $\mathcal{O}_{\mathsf{VerifyEnt}}$ and the verification function $\mathsf{VerifyAudA}$ running by auditors do the same check on the output tuple of $\mathsf{ShowupT}$ functions. Hence, the winning condition for $\mathcal{A}$ and for $\mathcal{B}$ are essentially the same. □

Appendix D. Card-whispering Protocol

Recall that to provide robust distribution (D3_{robust}) it is necessary to enable more than one member of a household to retrieve their entitlement. This is needed to support household members losing their cards or unable to attend the distribution due to sickness. Our solution achieves robustness by permitting that members of a household have “cloned” cards, i.e., cards that share the same state.

To create a card clone, registered household members need to bring their card to the registration station. Then, they run the following card-whispering protocol between a *parent card* – the card that the household member wants to clone, which contains the household secret and current state, and a *child card* – the card that will become a clone, which is initialized in the setup with the built-in shared keys and public parameters but does not contain any secret. The parent card and child card are inserted into dedicated devices. The protocol goes as follows:

- $\mathsf{CardWhispering}()$ The card whispering protocol is between a parent and a child card. The parent card takes as input its state $\mathsf{st}^P_T = (\epsilon_{\text{last}}, \mathsf{pk}, \mathsf{sk}, \mathsf{k}_H, \mathsf{ent}_H, \mathsf{v}_H)$, the client card has no input. The parent card sends st^P_T to the client. The client sets its own state st^C_T to equal st^P_T. The parent card does not modify its state.

After running the protocol, the child card will have the same state as the parent card (including the same secret of the household). We have described the protocol between two cards, but it can be run among any number of cards.

As explained in Section 4.2, to maintain privacy and security this protocol must be conducted in a controlled environment where the owner of the parent card can be authenticated and requires (1) that only the child card learns the household secret (otherwise privacy is at risk, see Section 4.2), and (2) that the child card is given to a member of the household (otherwise security is at risk, as illegitimate recipients may get the entitlement of the household).