PrivPass: Privacy-Preserving Digital Vaccine Passport

Thai Duong[*] Duong Hieu Phan[†] Ni Trieu[‡]

Abstract

Covid-19 pandemic with global lockdown has been caused severe social and economic problems. To re-open the economy while keeping the ability to control the spread of the disease, contact tracing and vaccine passport have been introduced. While the contact tracing has been well developed with a detailed security analysis via a long list of publications, vaccine passport has less attention in terms of formal problem definition, security requirement, and efficient construction.

In this work, we design a privacy-preserving digital vaccine passport system, PrivPass. By non-trivially combining different cryptographic primitives, we propose two efficient constructions that allow PrivPass work properly in a different environment while maintaining the user's privacy.

We show the feasibility of the proposed PrivPass system by estimating its performance. To verify whether a token of a passenger is valid, we expect that PrivPass can perform it in only 7 milliseconds with 480KB bandwidth requirement.

Keywords. Vaccine Passport, Anonymous Identity, Linkage Attack, PIR, Digital Signature

1 Introduction

After more than a year of social chaos with disease crisis, lockdown and restriction of freedom, we finally find ourselves in a more positive situation, because of the invention of the COVID-19 vaccines.

However, not everyone wants to be vaccinated, not everyone has the opportunity to be vaccinated soon due to the limited vaccine production capacity. It will therefore be risky to remove all restrictions, as the spread of the COVID-19 viruses is known to be very rapid. If a full reopening is infeasible at least in the near future, how can we target both a reasonable easing of restrictions and complete control of the pandemic? It is an intrinsic problem to be solved for any society as we want to preserve freedom of choice (everyone can be vaccinated or not), fight against all discrimination, and also return to normal life, where everyone has the right to freedom of movement.

Faced with such a challenge where goals cannot be achieved simultaneously, almost all countries have proposed the use of health status certificates providing digital proof that a person has been vaccinated, received a negative test result or recovered from COVID-19. Although motivated by the fact that the digital certificates should facilitate free movement, but shall not be a prerequisite for free movement, which is a fundamental human right, the discussion about such an application is emerging more and more. Risks of privacy breaches are becoming central issues in such mass development, not only for the current pandemic, but it may also set a standard for similar deployments in the future. This paper describes these issues and provides privacy protection options to resolve them.

[*]Google, thaidn@google.com
[†]LTCI, Telecom Paris, Institut Polytechnique de Paris, hieu.phan@telecom-paris.fr
[‡]Arizona State University, nitrieu@asu.edu

Now is the right time to consider privacy in such a vaccine passport rollout (we use the term vaccine passport to highlight the privacy issue, but this applies to all kinds of digital certificates). The issue of privacy in the vaccine passport is no less important than in contact tracing, but strangely enough, while many proposals from the scientific community were introduced last year for contact tracing, to the best of our knowledge, there is almost no published work on the privacy-preserving vaccine passports (at least no paper put on ePrint). It should also be noted that the incentives for the general public to use the vaccine passport are much higher than those to use the contact tracing app. Indeed, the use of the vaccine passport gives a direct and immediate impact to the passport holder to return to a normal life while the use of the contact tracing application serves rather to reduce the spread of virus circulation in a rather vague sense. This may explain why people were very suspicious of the privacy and effectiveness of contact tracing, but are more in favor of vaccine passports. Ipsos [IPS] surveyed over 21,000 people in 28 countries between March 26 and April 9, 2021 and found strong support of 78% for requiring travelers to carry COVID-19 vaccine passports. Interestingly, that same survey finds that, on average across 28 countries, just 50% are comfortable allowing their government to access their personal health information and 40% in the case of private companies.

Meanwhile, many governments and companies are deciding to use vaccine passports in the coming months. Let's take a look at some cases between countries with a tradition of fighting for privacy:

- European leaders have agreed to push ahead with plans for an EU-wide Digital Green Certificate as "a matter of urgency".
- Some European countries such as Denmark, Sweden, Iceland have launched their own vaccine passports. Denmark plans to use its "Coronapas" vaccine passport domestically and could also be used later as a tool for international travel.
- Estonia has launched distributed data exchange platform VaccineGuard to provide vaccination certificates in compliance with the EU's green certificate proposal. They also working with the World Health Organisation (WHO) to create a "smart yellow card" global vaccine certificate.
- UK ministers have recently confirmed that the main National Health Service (NHS) app will be soon updated to enable the public to show their COVID-19 vaccination or test results when traveling or attending public events.

Essentially, in the above solutions, after being vaccinated, a person will be given a certificate signed by the authority and can use it for their needs. It is specified in EU-wide Digital Green Certificate [EUC] that Digital Green Certificate contains a QR code with a digital signature to protect it against falsification. Then, when the certificate is checked, the QR code is scanned and the signature verified. It also ensures that, to protect users' privacy, the certificates will only include a limited set of information that cannot be retained by visited countries and all health data will remain with the member state that issued the certificate. However, it does mean that we have to trust the authority, if the QR code is marked then one can be traced for movement of a particular user. And even if the machine only scans the QR code, if it is hacked, all of a person's positions can be linked. The trust in authority and the linkability can pose serious privacy concerns. There is thus an urgent need to discuss privacy by design, not by trust, and propose methods that offer the highest level of privacy protection.

1.1 Our Contribution

In this work, we introduce PrivPass, a new system for privacy-preserving digital vaccine passport with strong security and lightweight cost. As a key primitive enabling PrivPass, we propose two cryptographic constructions that work properly with and without an Internet connection in the passport verification process. The proposed constructions are designed to be efficient on resource-constrained devices and scalable for a large number of users. In addition, the constructions allow us to meet all of our security and privacy goals. Thus, we believe that PrivPass is feasible in practice.

In summary, we make the following contributions:

1. We formally define the problem of digital vaccine passport, describe the desirable security and performance of an ideal vaccine passport scheme. To the best of our knowledge, this is the first work that formally studies this problem.

2. We proposed two efficient constructions for PrivPass: PIR-based construction for online verification and digital signature-based construction for offline verification. For each construction, we analyze its security property and complexity in terms of computation and communication.

3. Finally, we estimate the performance of PrivPass to show that the system feasible in practice. As desired, the (health) authority, the client (phone's holder), and the service (verifier) only need to perform a very lightweight computation. Concretely,

   - To generate/sign a valid vaccination certificate, the authority requires only 0.054ms per client who is vaccinated and linear in the number of clients.
   - The client's computation cost is constant per a redeemed token, which requires up to 13 milliseconds for a redeem process.
   - In our PIR-based construction, the service's computation cost is logarithmic in the number of valid tokens held by the cloud server. However, in our signature-based construction, it is constant per a redeemed token which takes only 155 milliseconds.

   The most computation cost is on the cloud server's side which can speed up with a stronger machine and multiple server/cores. Note that no information leaks to the cloud server, which allows PrivPass to outsource its computation without any privacy risk.

1.2 Related Work

Over the last year, there has been a long line of work [TPH+20, CGH+20, vABB+20, RPB20, Goo20, MMRV20, LAY+20, AIS20, LTKS20, CDF+20, ABB+20, CKL+20, CBB+20, TZBS20] on privacy-preserving contact tracing, a feasible method to control the spread of COVID-19. However, there has been very little work on digital vaccine passport, with current state-of-the-art vulnerable to user's privacy. To the best of our knowledge, all of them are under development without a formal description of their construction as well as their security guarantee. We review the Smart Health Cards Framework [1] which can be considered as the typical system for digital vaccine cards.

Smart Health Cards Framework is an open-source standard specified and adopted by hundreds of companies and organizations including Microsoft, IBM, and Mayo Clinic. The framework proposes a model that includes three parties:

- Issuer (e.g., a lab, pharmacy, healthcare provider, public health department, or immunization information system) generates verifiable tokens (also known as credentials)

[1] https://smarthealth.cards

- Holder/Client (e.g., a phone's holder) stores tokens and presents them at will
- Verifier/Service (e.g., a restaurant, an airline) receives tokens from the holder and ensures they are properly signed.

In the Smart Health Cards system, the holder has to disclose to the issuer their personally identifiable information (PII) (e.g., full name and date of birth) and immunization status (e.g., inoculation's location, date and time, vaccine type, lot number). From this set of information, the issuer might generate multiple tokens, each consists a subset of the holder's information and a digital signature. By selecting which token to share with a verifier, the holder can decide which granularity of information that they want to disclose to that verifier.

Although this framework aims to provide end-user privacy, it does not meet the desirable properties that we discuss in Section 2.4: the tokens are linkable and reusable. If a holder presents a token to two verifiers, the verifiers can reuse and link them up. If the verifiers collude with the issuer, they will be able to figure out the identity of the holder.

2 Problem Statement and Desirable Properties

In this section, we define the problem of digital vaccine passport that we intend to solve, describe the security definition and system's desirable properties in our PrivPass scheme.

2.1 Problem Definition

A digital vaccine passport is a mobile app that allows to verify whether the phone holder has been vaccinated for a certain disease (e.g. COVID-19). A digital vaccine passport system consists of three kinds of participants: a client $\mathcal{C}$, a health authority $\mathcal{A}$, and a service provider (or verifier) $\mathcal{S}$. When a client $\mathcal{C}$ is vaccinated, she receives a vaccination certificate σ from a health authority $\mathcal{A}$. The client $\mathcal{C}$ can use the certificate σ to prove a service provider that she is vaccinated without revealing her certificate σ. The proof process requires information from the health authority $\mathcal{A}$ who generated σ. In our PrivPass system, we rely on an untrusted cloud server $\mathcal{H}$ who computes a workload of the health authority $\mathcal{A}$ on their behave to improve the system's efficiency while revealing nothing to the cloud server $\mathcal{H}$.

2.2 Security Definition

In the vaccine passport system, we consider four types of participants: a client (or a phone's holder) $\mathcal{C}$, an authority server $\mathcal{A}$, a cloud server $\mathcal{H}$, and a service (or verifier) $\mathcal{S}$. All participants have agreed upon a single functionality (i.e. vaccine passport verification) to compute and have also consented to give the final result to some particular party. At the end of the computation, nothing is revealed by the computational process except the final output.

In the ideal execution, the participants interact with a trusted party that evaluates the function in the presence of a simulator that corrupts the same subset of participants. In the real-world execution, the participants often perform the protocol's execution in the presence of an adversary who corrupts a subset of the participants. Typicaly, there are two adversarial models and two models of collusion.

- Adversarial model: A *semi-honest* adversary is assumed to follow the protocol but attempts to obtain additional information from the execution transcript. A *malicious* adversary allows to apply any arbitrary polynomial-time strategy so that he can deviate from the protocol and try to learn extra information as much as he can.

- Collusion security: A *non-colluding* model is considered as independent adversaries, each observing the view of each independent dishonest party. The model is secure if the individual distribution of each view can be simulated. A *colluding* model is considered as a single monolithic adversary observing the possibility of collusion between the dishonest parties. The model is secure if the joint distribution of those views can be simulated.

For simplicity, we assume there is an authenticated secure channel (e.g., with TLS) between each pair of participants. In this work, we consider the malicious setting and colluding model. The privacy of the users will be guaranteed if the adversary can corrupt parties but as long as the authority server $\mathcal{A}$ and the service $\mathcal{S}$ are not corrupted. We describe more detail on the security of our system in Section 4. Following security definitions of [Ode09, KMR12], we formally present the security definition considered in this work.

Real-world execution. The real-world execution of protocol Π takes place between a set of users $(\mathcal{C}_1, \ldots, \mathcal{C}_n)$, an authority server $\mathcal{A}$, a cloud server $\mathcal{H}$, a set of services $(\mathcal{S}_1, \ldots, \mathcal{S}_N)$, and a set of adversaries $(\mathsf{Adv}_1, \ldots, \mathsf{Adv}_m)$. Let H denote the honest participants, I denote the set of corrupted and non-colluding participants, and C denote the set of corrupted and colluding participants.

At the beginning of the execution, each participant receives its input x_i, an auxiliary input a_i, and random tape r_i. These values x_i, a_i can be empty. Each adversary $\mathsf{Adv}_{i \in [m-1]}$ receives an index $i \in I$ that indicates the party it corrupts. The adversary Adv_m receives C indicating the set of parties it corrupts.

For all $i \in H$, let out_i denote the output of honest party, let out'_i denote the view of corrupted party for $i \in I \cup C$ during the execution of Π. The i^{th} partial output of a real-world execution of Π between participants in the presence of adversaries $\mathsf{Adv} = (\mathsf{Adv}_1, \ldots, \mathsf{Adv}_m)$ is defined as

$$\texttt{real}^{i}_{\Pi,\mathsf{Adv},I,C,y_i,r_i}(k, x_i) \overset{\mathsf{def}}{=} \{\mathsf{out}_j \mid j \in H\} \cup \mathsf{out}'_i$$

Ideal-world execution. All the parties interact with a trusted party that evaluates a function f in the ideal-world execution. Similar to the real-world execution, each participant receives its input x_i, an auxiliary input y_i, and random tape r_i at the beginning of the ideal execution. The values x_i, y_i can be empty. Each participant sends their input x'_i to the trusted party, where x'_i is equal to x_i if this user is semi-honest, and is an arbitrary value if he is malicious. If any honest participant sends an abort message $(\perp)$, the trusted party returns $\perp$. Otherwise, the trusted party then returns $f(x'_1, \ldots, x'_n)$ to some particular parties as agreed before.

For all $i \in H$, let out_i denote the output returned to the honest participant by the trusted party, and let out'_i denote some value output by corrupted participant $i \in I \cup C$. The i^{th} partial output of a ideal-world execution of Π between participants in the presence of independent simulators $\mathsf{Sim} = (\mathsf{Sim}_1, \ldots, \mathsf{Sim}_m)$ is defined as

$$\texttt{ideal}^{i}_{\Pi,\mathsf{Sim},I,C,z_i,r_i}(k, x_i) \overset{\mathsf{def}}{=} \{\mathsf{out}_j \mid j \in H\} \cup \mathsf{out}'_i$$

[Ode09, KMR12] (Security) Suppose f is a deterministic-time n-party functionality, and Π is the protocol. Let x_i be the parties' respective private inputs to the protocol. Let $I \in [N]$ denote the set of corrupted and non-colluding parties and $C \in [N]$ denote the set of corrupted and colluding parties. We say that protocol $\Pi(I, C)$ securely computes deterministic functionality f with abort in the presence of adversaries $\mathsf{Adv} = (\mathsf{Adv}_1, \ldots, \mathsf{Adv}_m)$ if there exist probabilistic polynomial-time simulators $\mathsf{Sim}_{i \in m}$ for $m < n$ such that for all $\bar{x}, \bar{y}, \bar{r} \leftarrow \{0, 1\}^{\star}$, and for all $i \in [m]$,

$$\{\mathtt{real}^{i}_{\Pi,\mathsf{Adv},I,C,\bar{y},\bar{r}}(k,\bar{x})\overset{c}{\equiv}\{\mathtt{ideal}^{i}_{\Pi,\mathsf{Sim},I,C,\bar{y},\bar{r}}(k,\bar{x})\}$$

Where $\mathsf{Sim} = (\mathsf{Sim}_1, \ldots, \mathsf{Sim}_m)$ and $\mathsf{Sim} = \mathsf{Sim}_i(\mathsf{Adv}_i)$

2.3 Desirable Security

We describe security and privacy requirements of the privacy-preserving digital vaccine passport system. One of the main desirable properties is that the system would make honest user's actions are perfectly indistinguishable from the actions of all other honest users as well as other participants (e.g. authority server $\mathcal{A}$, cloud server $\mathcal{H}$, and service $\mathcal{S}$). Thus, an ideal digital vaccine passport system would guarantee that executing the system in the real model is equivalent to executing this system in an ideal model with a trusted party. This requirement is formally presented in Definition 2.2.

Based on the definition, we consider the following security and privacy desirable properties in the context of the vaccine passport system.

- **Anonymous Identity**: the real (or underlying) identity of a client $\mathcal{C}$ is not revealed to the untrusted cloud server $\mathcal{H}$. The identity is also unknown to the service $\mathcal{S}$ if the client $\mathcal{C}$ does not require to present its identity to him. Note that our PrivPass system does not maintain anonymity for clients who explicitly or implicitly publish their identifiable information. In addition, the authority $\mathcal{A}$ is allowed to know only the identity of a client $\mathcal{C}$ who has been vaccinated.

- **Token Unlinkability**: We consider valid tokens which are generated from the same vaccination certificate σ, and may be redeemed at several services $\mathcal{S}$. The tokens of the same client cannot be linked together by any participant. Clearly, the client solely knows their own set of tokens which remains unlinkable during the whole process unless the clients explicitly publish this information. Our PrivPass does not prevent token unlinkability when a set of services collude with the authority server $\mathcal{A}$.

- **Token Unforgeability**: All vaccination tokens are unforgeable. A client should not be able to compute a valid token unless it was generated by the valid vaccination certificate σ of that client. A client should not be able to compute a valid token generated by a vaccination certificate σ of another client. Clients can not redeem forgeable tokens. They will be caught if the redeemed token is invalid.

- **Token Unreusability**: Each valid token can be only used one time. When a token is redeemed, it is immediately deleted from the client's device. A client cannot reuse redeemed tokens. All involved participants (including the services) also cannot reuse redeemed tokens.

In the following sections, we will analyze our privacy-preserving vaccine passport constructions that satisfy all of the security and privacy requirements listed above.

2.4 Desirable Performance

Besides the desirable security and privacy, an ideal privacy-preserving digital vaccine passport system should have requirements for the its performance. We consider the following desirable performance properties:

- **Efficiency**: The digital vaccine passport system should be able to process a verification computation within a few seconds, and scalable to a large number of users. In addition, participants, especially the authority $\mathcal{A}$ and the client $\mathcal{C}$, should do lightweight work.
- **Flexibility:** In some places where the client's ID should be collected by a service (e.g. at the airport), the vaccine passport system cannot avoid anonymous identity. The system can be flexible to provide a trade-off between performance and security in this case. Similarly, the system can enable cost savings for other places where ID is not required.
- **Online/Offline Redeem**: In practice, a service might have a slow network or cannot connect to the internet during a verification process, the system should work properly in different situation of the network connection (e.g. support both online and offline redeem process).

In Section 5, we estimate the computation and communication of our proposed constructions. We show that our system PrivPass satisfies the desirable performance.

3 Cryptographic Preliminaries

This section introduces the notations and cryptographic primitives used in the later sections.

3.1 Notation

For $n \in \mathbb{N}$, we write $[n]$ to denote the set of integers $\{1, \ldots, n\}$. We use '$||$' to denote string concatenation. In this work, the computational and statistical security parameters are denoted by κ, λ, respectively.

3.2 Diffie–Hellman assumption

Our PrivPass system is essentially based on the CDH or DDH assumption in a cyclic group. These assumptions are defined as below.

[DH06] Let $\mathbb{G}(\kappa)$ be a group family parameterized by security parameter λ. For every probabilistic adversary Adv that runs in polynomial time in λ, we define the advantage of Adv to be:

$$\Pr[\mathsf{Adv}(g, g^a, g^b, g^{ab}) = 1] - \Pr[\mathsf{Adv}(g, g^a, g^b, g^c) = 1]$$

Where the probability is over a random choice $\mathbb{G}$ from $\mathbb{G}(\lambda)$, random generator g of $\mathbb{G}$, random $a, b, c \in [|G|]$ and the randomness of Adv. We say that the Decisional Diffie–Hellman assumption holds for $\mathbb{G}$ if for every such Adv, there exists a negligible function ϵ such that the advantage of Adv is bounded by $\epsilon(\lambda)$.

Let $\mathbb{G}$ be a cyclic group of order N, and let g be its generator. The Computational Diffie-Hellman (CDH) problem is hard in $\mathbb{G}$ if no efficient algorithm given (g, g^a, g^b) can compute g^{ab}.

3.3 Private Information Retrieval

Private Information Retrieval (PIR) [CMS99] allows a client to request information from one or multiple servers in a such way that the servers do not know which information the client queried. More formally, the server(s) hold a database DB of N strings, and the client wishes to read data record $DB[i]$ without revealing i. In general, PIR construction [DC14, GR05, ACLS18, ALP+19] consists of three procedures below:

- $\mathsf{PIR.Gen}(\kappa) \rightarrow (pk, sk)$: a randomized algorithm that takes a security parameter and generates an (additively homomorphic) public and private key pair (pk, sk).
- $\mathsf{PIR.Query}(pk, i) \rightarrow k$: a randomized algorithm that takes index $i \in [N]$ and public key pk as input and outputs an evaluated key k.
- $\mathsf{PIR.Answer}(pk, k, DB) \rightarrow c$: a determinated algorithm that takes an evaluated key k, public key pk, and a database DB as input, returns c.
- $\mathsf{PIR.Extract}(sk, c) \rightarrow DB[i]$: takes a secret key sk and answer c as input, returns d.

A PIR construction is correct if and only if $d = DB[i]$. We say that PIR is (symmetric) secure if an evaluated key k reveals nothing about the index i and the answer c reveals nothing about other database record $DB[j], j \in [N], j \neq i$. Recent PIR [AMBFK16, ACLS18] is significantly faster with logarithmic communication cost in the database size.

Keyword-PIR. A variant of PIR called keyword-PIR was introduced by Chor, et al. [CGN98]. In keyword-PIR, the client has an item x, the server has a set S, and the client learns whether $x \in DB$. This variant of PIR has many real-world applications. For example, Google has recently used keyword-PIR for the password checkup problem [ALP+19]. In that application, a client aims to check whether their password is in breached data, without revealing the password itself to Google. The most efficient keyword PIR [ALP+19, DRRT18] is implemented using bucketing with Cuckoo hashing [DC14]. In this paper, we are interested in Keyword PIR based on 1-server PIR [ACLS18, ALP+19], but our protocol can use multiple-server PIR [DG16, BGI15, BGI16, CK20] to speed up the system's performance.

Similar to the traditional PIR, a keyword-PIR construction [DC14, ACLS18, ALP+19] consists of the same four procedures. Different from PIR, in keyword PIR, the $\mathsf{PIR.Query}(pk, x)$ takes a keyword x as input , and the $\mathsf{PIR.Extract}$ returns a bit d indicated whether x is in the server's dabase DB. Using hashing technique [ACLS18, ALP+19], the keyword-PIR has a similar computation and communication as the traditional PIR.

The work of Angel *et al* ([ACLS18, Figure 5]) shows that a PIR query on a database of size 2^{20} costs about 7.62 milliseconds on the client's side and 80 milliseconds on the server's side (online time). The query requires 480KB bandwidth communication.

PIR-with-Default. Another variant of PIR called PIR-with-Default was introduced by Lepoint, et al. [LPR+20]. In PIR-with-Default, the server has a set of key-value pairs $P = \{(x_1, v_1), \dots, (y_n, v_n)\}$ with distinct y_i and pseudo-random v_i, and a default (pseudo-random) value w. The client inputs an item x, and obtains v_i if $x = v_i$, and w otherwise. The default value w has to be refreshed for each query. This variant of PIR has been used in applications of private join and compute.

Similar to the keyword-PIR, a PIR-with-Default construction also consists of the same procedures. Different from the keyword-PIR, in PIR-with-Default, the $\mathsf{PIR.Answer}(pk, k, P, w)$ takes P and w as input, and the $\mathsf{PIR.Extract}$ returns a value v.

The PIR-with-Default protocol [LPR+20] is efficient which can enable 2^8 PIR with default lookups on a database of size 2^{20} with the communication of 7MB and the online computation of 2.43 milliseconds.

3.4 Private Matching

A Private Matching (PM) is a 2-party protocol in which the sender with input string m_0 interacts with a receiver with input string m_1 in the following way. The receiver learns a bit indicating whether $m_0 = m_1$ and nothing else, while the sender learns nothing about m_1.

PARAMETERS: Two parties: sender and receiver

FUNCTIONALITY:

- Wait for input $m_0 \in \{0,1\}^*$ from the sender.
- Wait for input $m_1 \in \{0,1\}^*$ from the receiver.
- Give the receiver output 1 if $m_0 = m_1$ and 0 otherwise.

PROTOCOL:

- The receiver chooses a random exponent r, computes $u \leftarrow H(m_1)^r$ and sends it to the sender
- The sender chooses a random exponent k, computes $v \leftarrow u^k$ and sends it to the receiver
- The sender computes $w \leftarrow H(m_0)^k$ sends it to the receiver
- The receiver output 1 if $v^{1/r} = w$ and 0 otherwise.

Figure 1: The Private Matching functionality

To the best of our knowledge, PM was first introduced by the work [Mea86, FNW96]. PM is heavily used in private set intersection (PSI) protocols [FNP04, PRTY20], pattern matching [KRT18], password checkup [Goo19], and private contact tracing applications related to COVID-19 [TSS+20, BBV+20, DPT20, DIL+20]. Executing a batch of PM instances is very efficient based on OT extension. For example, [KKRT16] shows that the amortized cost of each PM instance with unbounded input domain $\{0,1\}^*$ is only a few symmetric-key operations and 488 bits in communication.

Our protocol requires to execute a PM instance at a time. Thus, the construction of [KKRT16] is not quite suitable for us. In our PrivPass system, we use the DH-based PM scheme which works as follows.

The receiver computes $u \leftarrow H(m_1)^r$ for some random, secret exponent r, and sends it to the sender. The sender raises u to the random secret k power and obtains u^k. The sender then sends the result to the receiver who can then raise the result to the $1/r$ power to obtain $H(m_1)^k$. The sender now sends $H(m_0)^k$ to the receiver, who can check whether $H(m_0)^k = H(m_1)^k$. In a case that $m_1 \neq m_0$, the receiver learns nothing about m_1 from $H(m_1)^k$. It relies on Diffie–Hellman assumption mentioned in Section 3.2.

We describe the ideal functionality and the DH-based construction of PM in Figure 1. The computation and communication cost of PM is 3 exponentiations and 3 group elements, respectively.

3.5 Randomizable Signature Scheme

Digital signature [Mer90, Sch90] with advanced features such as randomizability has been used as a very important building block. This feature enables anyone to derive, from a valid signature σ, a new valid signature $\sigma^\star$ on the same message. The randomizability should guarantee that these two signatures are unlikable, even for the signer. The first construction achieving such property was proposed by Camenisch and Lysyanskaya [CL04] and recently improved by Pointcheval and Sanders [PS16, PS18]. Randomizable signature schemes are perfectly suitable for our setting because we need to preserve the privacy of users even if the authority - who provides signed certificates - is corrupted.

Bilinear Group Setting A *bilinear group generator* $\mathcal{G}$ is an algorithm that takes as input a security parameter λ and outputs a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ such that $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$ are cyclic groups of prime order p (a λ-bit prime integer), and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an *admissible pairing*:

- e is bilinear: for all $a, b \in \mathbb{Z}_p, e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- e is efficiently computable (in polynomial-time in λ);
- e is non-degenerate: $e(g_1, g_2) \neq 1$.

Furthermore, the bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e)$ is said *asymmetric* when $\mathbb{G}_1 \neq \mathbb{G}_2$.

There are three types of pairings: in type 1, $\mathbb{G}_1 = \mathbb{G}_2$; in type 2, e is asymmetric but there exists an efficient homomorphism from $\mathbb{G}_2 \to \mathbb{G}_1$ while no efficient one exists in the other direction; in type 3, e is asymmetric and there is no efficiently computable homomorphism exists between $\mathbb{G}_1$ and $\mathbb{G}_2$. The CL signature uses pairings of type 1, while Pointcheval and Sanders proposed a new signature scheme in type 3, with the same features as CL signatures, but with a constant size signature. The unlinkability of the Pointcheval-Sanders scheme is based on the DDH assumption in G_1 and the unforgeability is based on rather a complex assumption defined in [PS16]. As type 3 offer the best performances and it is important for us to optimize the bandwidth in the large-scale application of vaccine passport, we rely on the Pointcheval-Sanders signature.

4 Digital Vaccine Passport Constructions

We begin with describing the overview of our PrivPass system. We then present two cryptographic constructions: one is for online verification, and another is for offline verification.

4.1 System Overview

The goal of a digital vaccine passport is to prove a service whether the phone holder (e.g. client) has been vaccinated for a certain disease. In this section, we describe the overview of our proposed PrivPass system which consists of three main procedures.

- `RegistrationRequest`$(\kappa, inf) \to \sigma$: A client $\mathcal{C}$ uses the `RegistrationRequest` function to submit a certificate request to a health authority $\mathcal{A}$. The function takes a security parameter κ and the client's information inf as input. The health authority $\mathcal{A}$ checks whether the client has been vaccinated. If yes, the health authority returns $\mathcal{A}$ to the user a valid vaccination certificate σ. In addition, the health authority sends some anonymous information to the cloud server $\mathcal{H}$.
- `TokenGeneration`$(\sigma, n) \to \{tok_0, ..., tok_n\}$: A client $\mathcal{C}$ uses the `TokenGeneration` function and optionally interacts with the cloud server $\mathcal{H}$ to generate a list of n vaccination tokens. The function takes a vaccination certificate σ and the number n as input, and outputs n tokens.
- `TokenRedeem`$(tok_t, [inf]) \to \{0, 1\}$: A client $\mathcal{C}$ uses the `TokenRedeem` function to redeem a token tok_t at the time t. The function takes as input the token tok_t and the client's information inf if required. A service provider (verifier) $\mathcal{S}$ optionally interacts with the cloud server $\mathcal{H}$ to check whether tok_t is valid and belongs to the holder of the code. The output might return to the client $\mathcal{C}$. The token tok_i becomes invalid after redeem.

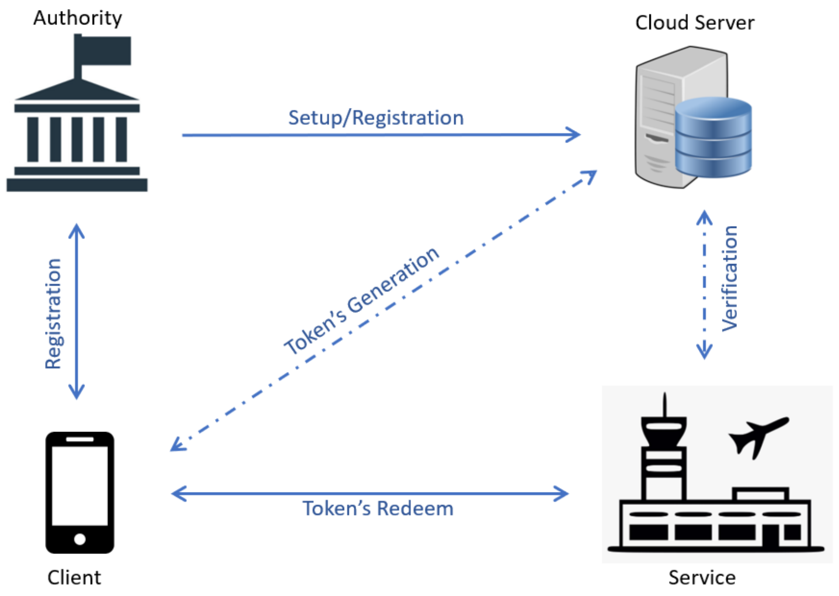

Figure 2: The Overview of our PrivPass System which consists of three main phases: Registration, Token's Generation, Token's Redeem. The solid and dashed lines show the required and optimal communication/connection between the participants, respectively.

4.2 PIR-based Construction (Online Verification)

We begin with the vaccine passport construction where a service requires to be online for verifying whether a token is valid. The construction heavily relies on different PIR variants.

4.2.1 Technical Overview.

At a starting point, we consider a blueprint solution in which a client $\mathcal{C}$ obtains a vaccination certificate σ from the authority $\mathcal{A}$ upon having vaccinated. Whenever the client $\mathcal{C}$ visits a place and wants to show that she was vaccinated, $\mathcal{C}$ presents σ to a service that optimally communicates with the authority to verify the σ's validation in a privacy-preserving manner. A similar variant of the blueprint solution is currently implemented by Smart Health Cards Framework that we discussed in Section 1.2

While the above solution can implement a basic functionality of a digital vaccine passport, it is vulnerable to the desirable security described in Section 2.3. For example, it allows several corrupted services to link the tokens of the same client. Moreover, the blueprint solution also requires the authority to perform a heavy secure computation for the token's verification, which is against the desirable performance described in Section 2.4.

To fulfill the desirable security of a vaccine passport system, we modify the blueprint construction so that the client $\mathcal{C}$ can prove the service $\mathcal{S}$ that she was vaccinated while keeping the vaccination certificate σ secret from him. Concretely, redeem tokens are generated by computing PRF $tok \leftarrow F(\sigma, t)$, where t indicates the time of the token redeem. By doing so, all generated tokens are unlinkable due to the underlying PRF, and each token can be redeemed one at a time. For some places, each token can be attached with an encryption of the client's ID to avoid other clients from using a valid token of a particular client.

For the desirable performance of the system, we observe that the authority $\mathcal{A}$ indeed can delegate their computation to an untrusted cloud server $\mathcal{H}$. If the vaccination certificate σ is computed from a random key r and the client's information, revealing σ to $\mathcal{H}$ leaks nothing as long as r is secret and known by the authority $\mathcal{A}$ only. In our construction, $\sigma = F(r, I_2||...||I_n)$ where r is randomly chosen by the authority, I_1 represent the client's ID, other $I_i \in \{0,1\}^\star$ represents additional information about the vaccine taken by the client such as "type of vaccine", "effective date", etc. Having the vaccination certificate σ, the cloud server $\mathcal{H}$ can generate a list T of valid tokens $tok \leftarrow F(\sigma, t)$. The authority $\mathcal{A}$ also sends the cloud server $\mathcal{H}$ the group element $m = g^{H(I_1)}$ for anonymizing the user's identification. Based on Diffie–Hellman's assumption, m reveals nothing about the actual client's ID.

To verify token's validation, the service $\mathcal{S}$ with a token tok obtained from the client, and wishes to check whether tok is in the list of valid tokens hold by the cloud server $\mathcal{H}$. Clearly, it can be implemented using Keyword-PIR described in Section 3.3. In particular, the service $\mathcal{S}$ sends a PIR request as $\mathsf{PIR.Query}(pk, tok) \to k$, and obtains $\mathsf{PIR.Answer}(pk, k, T) \to c$ from cloud server $\mathcal{H}$. Using $\mathsf{PIR.Extract}(sk, c) \to \{0,1\}$, the service $\mathcal{S}$ leans whether the token tok is valid. Here, the public-private key pair pk, sk are generated from $\mathsf{PIR.Gen}(\kappa) \to (pk, sk)$ by the service $\mathcal{S}$.

Depending on whether the client's ID is compulsorily collected by the service, we consider two following cases. In the first case where the service $\mathcal{S}$ (e.g. an airline company) has the client's identify I_1 in the clear, $\mathcal{S}$ can compute $m = g^{H(I_1)}$ and attach it with the token as $tok||m$ before sending a PIR request. Similarly, the cloud server modifies T to have a set of $tok||m$ before returning a PIR answer to the service.

In the second case where the service does not allow to collect the client's ID, our construction relies on PIR-with-Default and Private Matching. More precisely, the cloud server $\mathcal{H}$ makes a set of pairs (tok, m), and allows the service $\mathcal{S}$ to retrieve either m or a random default value via PIR-with-Default. Next, the service $\mathcal{S}$ with the obtained PIR output and the client $\mathcal{C}$ with m involve an instance of private matching to learn whether the client redeemed the valid token generated from her vaccination certificate σ.

4.2.2 Construction.

Our PIR-based vaccine passport construction is presented in Figure 3. It closely follows the technical overview described above. We divide our construction into three phases to align with the system overview described in Section 4.1. The first phase includes the vaccination certificate computation and distribution by the authority $\mathcal{A}$. In the second phase, each client $\mathcal{C}$ and the cloud server $\mathcal{H}$ locally generate valid tokens from the obtained vaccination certificate. The last phase is for the token's redeem process which involves the computation of all participants except the authority $\mathcal{A}$ (who only involves in the first phase).

In Step (III,3), the client $\mathcal{C}$ can indeed compute $m' = g^{H(I_1)}$ and gives the exponentiation to the service $\mathcal{S}$ who then involves Keyword-PIR with the could service (similar to Step (III,2)). However, this modification might not completely hide the client's information from the service $\mathcal{S}$. For example, if later $\mathcal{S}$ might know I_1 by chance, he can de-identify the client.

The PIR-with-Default can be replaced by a cheaper cryptographic primitive, Oblivious Programmable PRF [KMP+17, PSTY19] (OPPRF). In the OPPRF construction, the default value w is chosen by the functionality, which is different from PIR-with-Default where w is a part of the sender's input. However, using PIR-with-Default with a zero string w allows the service $\mathcal{S}$ to terminate the computation earlier.

It is easy to see that correctness is obvious from the definitions of PIR variants, private matching, and Diffie–Hellman's assumption.

PARAMETERS:

- A client $\mathcal{C}$, an authority $\mathcal{A}$, a service $\mathcal{S}$, and a cloud server $\mathcal{H}$.
- A cyclic group $G =< g >$ of prime order p
- A PRF function $F : (\{0,1\}^\kappa, \{0,1\}^\star) \rightarrow \{0,1\}^\kappa$
- A hash function $H : \{0,1\}^\star \rightarrow \{0,1\}^\kappa$
- A Keyword-PIR, PIR-with-Default, and Private Matching primitives.

INPUTS: When a client $\mathcal{C}$ is vaccinated, it is associated with an information vector $\vec{I} = \{I_1, \dots, I_n\} \in (\{0,1\}^\star)^n$, where I_1 represent the client's ID, other $I_i \in \{0,1\}^\star$ represents information about the vaccine taken by the client such as "type of vaccine", "effective date", etc.

PROTOCOL:

I. **Registration Phase**
   - The client $\mathcal{C}$ sends $\vec{I} = \{I_1, \dots, I_n\}$ to the authority $\mathcal{A}$
   - $\mathcal{A}$ chooses a random value r, computes $\sigma = F(r, I_2||...||I_n)$ and $m = g^{H(I_1)}$
   - $\mathcal{A}$ distributes σ to both $\mathcal{C}$ and $\mathcal{H}$. Besides, $\mathcal{A}$ sends m to $\mathcal{H}$.

II. **Tokens Generation Phase**
   - For each pair (σ, m) received from $\mathcal{A}$, the cloud server $\mathcal{H}$ computes a set T_m of valid tokens $F(\sigma, t)$, where t indicates the time of redeem (e.g. every 15 minutes).
   - $\mathcal{C}$ generates a list of tokens $tok'_t = F(\sigma, t)$, where t is the redeem time.

III. **Token's Redeem Phase**: At the time t,
   1. If presenting the client's identity I_1 is not required (light verification):
      - $\mathcal{C}$ sends tok'_t to $\mathcal{S}$
      - $\mathcal{S}$ and $\mathcal{H}$ involve a keyword-PIR instance:
        * $\mathcal{S}$ sends a PIR query $\mathsf{PIR.Query}(pk, tok'_t) \rightarrow k$ to $\mathcal{H}$
        * $\mathcal{H}$ replies $\mathsf{PIR.Answer}(pk, k, D) \rightarrow c$ to $\mathcal{S}$ where D is a set of tok, for all $tok \in T_m$
        * $\mathcal{S}$ outputs $\mathsf{PIR.Extract}(sk, c)$
   2. If the client's identity I_1 is collected by the service $\mathcal{S}$
      - $\mathcal{C}$ sends (tok'_t, I_1) to $\mathcal{S}$ who computes $m' = g^{H(I_1)}$
      - $\mathcal{S}$ and $\mathcal{H}$ involve a keyword-PIR instance:
        * $\mathcal{S}$ sends a PIR query $\mathsf{PIR.Query}(pk, tok'_t||m') \rightarrow k$ to $\mathcal{H}$
        * $\mathcal{H}$ replies $\mathsf{PIR.Answer}(pk, k, D) \rightarrow c$ to $\mathcal{S}$ where D is a set of $tok||m$, for all $tok \in T_m$
        * $\mathcal{S}$ outputs $\mathsf{PIR.Extract}(sk, c)$
   3. If presenting the client's identity I_1 is required, but the service does not allow to collect the client's identity I_1
      - $\mathcal{C}$ sends tok'_t to $\mathcal{S}$
      - $\mathcal{S}$ and $\mathcal{H}$ involve a PIR-with-Default instance:
        * $\mathcal{S}$ sends a PIR query $\mathsf{PIR.Query}(pk, tok'_t) \rightarrow k$ to $\mathcal{H}$
        * $\mathcal{H}$ replies $\mathsf{PIR.Answer}(pk, k, D, w) \rightarrow c$ to $\mathcal{S}$ where D is a set of pairs (tok, m), for all $tok \in T_m$, and w is a zero string.
        * $\mathcal{S}$ computes $\mathsf{PIR.Extract}(sk, c) \rightarrow v$
      - If v is a zero string, the $\mathcal{S}$ outputs 0 (i.e. tok'_t is invalid). Otherwise, the service $\mathcal{S}$ and the client $\mathcal{C}$ involve a private matching instance:
        * $\mathcal{C}$ acts as a sender with input $m' = g^{H(I_1)}$.
        * $\mathcal{S}$ acts as a receiver with input v and output whether $v = m'$.

Figure 3: Our PIR-based Vaccine Passport Construction.

4.2.3 Security.

We analyze the security of the proposed PIR-based construction according to our desirable security and privacy of a digital vaccine passport system.

Anonymous Identity. By design, we allow the authority know the client's identity. We turn to show that the identity is not revealed to the cloud server. In some cases, it is also hidden from the service.

We begin with the view of the corrupt cloud server $\mathcal{H}$ which consists of the vaccination certificate σ, the exponentiation $m = g^{H(I_1)}$, and PIR's transcripts. We assume that the cloud server $\mathcal{H}$ does not collude with the authority $\mathcal{A}$, thus σ looks random to $\mathcal{H}$ as he does not know the authority's secret value r. Our construction relies on the difficulty in solving the discrete log problem (i.e. Diffie–Hellman's assumption). Thus, given $m = g^{H(I_1)}$, the could server $\mathcal{H}$ cannot recover the client's identity I_1.

We first consider a case where the service does not allow to collect the client's identity I_1, but presenting the ID is compulsory (ref. Step IV,3 in Figure 3). The view of the corrupt service $\mathcal{S}$ consists of the redeemed token tok'_t, PIR's and private matching transcripts. The tokens tok'_t is generated from the PRF key σ which is unknown to $\mathcal{S}$. Thus, tok'_t looks random to him. Because of PIR and private matching pseudorandomness property, the real identity of the client is protected. For a case where presenting the client's identity is not required, the security analysis on anonymous identity is similar.

Token Unlinkability. One of the important properties of a vaccine passport system is token unlinkability as it prevents learning the travel history of a particular user, so preventing an attacker from recovering the user's personality. The following section discusses how PrivPass provides unlinkability. Particularly, we demonstrate the difficulty for an attacker to link multiple individual tokens together, at each relevant step of the protocol. Note that attacks by matching IP addresses of the clients are beyond this work. We assume that the clients are responsible for using secure channels during communications with the service providers.

- Phase 1. Registration Phase: During this registration phase, client and the cloud server communicate with the authority $\mathcal{A}$ and obtain a vaccination certificate σ and a value m from $\mathcal{A}$. First, it is not practical for an attacker to recover the client's information $\{I_2, \ldots, I_n\}$ from the PRF value without the secret value r of the authority (unless the authority $\mathcal{A}$ is compromised). Second, the value m looks random to the attacker due to Diffie–Hellman's assumption.

- Phase 2. Token Generation Phase: This phase is locally computed by each individual client and the cloud server. Thus, there is no communication/computation between the participants, so no information is leaked. However, if an attacker attacks the cloud server, he knows which tokens are generated from the same vaccination certificate σ, but he does not know at which places the tokens are redeemed (see Phase 3).

- Phase 3. Token's Redeem Phase: During this phase, if an attacker controls a subset of service providers $\mathcal{S}$ who collected *the client's IDs*. PrivPass cannot provide unlinkability in this case. For other cases, if the attacker collects a list of redeemed tokens from different service providers $\mathcal{S}$. All redeemed tokens are random even they might be generated from the same key σ. In addition, each token is for one-time use only. Thus, no linkability can be made between tokens.

In case that the attacker controls the cloud server, he also learns nothing due to the PIR ideal functionality. The cloud server does not know which tokens were redeemed at which places.

Token Unforgeability. According to our construction, if a token *tok* is not generated from a valid vaccination certificate, the service is able to detect this event via PIR. Recall that the cloud server $\mathcal{H}$ has a set of valid tokens, PIR functionality allows the service to check whether *tok* is in the $\mathcal{H}$'s database. Moreover, PIR-with-default allows $\mathcal{S}$ to retrieve anonymous information of the client's identity m. Private matching between $\mathcal{S}$ and $\mathcal{C}$ prevents an attacker to redeem tokens of another client.

Token Unreusability. Each token is associated with a redeem time which eliminates an attacker to reuse the token later. However, PrivPass cannot prevent the attacker to redeem the same token at the same time at two different service providers unless the ID presentation is required. Therefore, we trust an end-user device to delete the token after it is redeemed. To remove the trusted assumption, we might use a secure deletion to obliviously remove the redeemed tokens from the cloud server's database T. However, the cloud server can observe which token was deleted. To handle this issue, one might rely on two or multiple cloud servers, each holds secret shares of T. All the shares must be re-randomized after an oblivious deletion event is executed.

Finally, we present the security of our PIR-based construction by the following theorem. The security proof straightforwardly follows from the security of its building blocks and the security discussion above. Thus, we omit its proof.

Given the Keyword-PIR, PIR-with-Defalt, Private Matching functionalities described in Section 3, the PIR-based construction of Figure 3 securely implements the digitial vaccine passport described in Section 2 in semi-honest setting.

4.2.4 Complexity.

We begin by the analysis of the computational complexity. As desired, the authority only needs to perform one PRF (e.g, AES), and one exponentiation per client who was vaccinated.

The cloud server $\mathcal{H}$ requires to perform N AES calls to generate the set T_m. N can set to be 80 if assuming that a token is generated every 15 minutes for approximately 20 hours a day. The cloud server $\mathcal{H}$ also involves PIR with the service in Phase 3. Denote the computational cost of PIR as $|\texttt{PIR}|$ which is $O(Nn)$, where n is the number of vaccinated clients. The cloud server computational complexity is $N + |\texttt{PIR}|$. The client requires to compute N AES instances and one exponentiation in Phase 2. In the token redeem phase, she might require to perform private matching with the service. This computation occurs another two exponentiations as described in Section 3.4. The computation on the service's side consists of PIR, private matching if the client requires to present his ID, but does not allow the service to collect it, and one exponentiation if the service can collect the ID.

For the communication complexity, the authority sends a κ-bit σ to the client and sends a 3κ-bits $\sigma||m$ to the cloud server. The client sends the service a κ-bit token together with 2κ-bits m if required. In addition, all participants except the authority send and receive transcript/randomness from PIR or private matching execution.

4.3 Digital signature-based Construction (Offline Verification)

We now deal with the vaccine passport construction where a service does not need to be online for verifying whether a token is valid. The construction relies on randomized signatures and signatures on committed values.

4.3.1 Technical Overview.

Again, at a starting point, a client $\mathcal{C}$ obtains a vaccination certificate σ from the authority upon having vaccinated. In the PIR-based construction, the verification of a valid token is carried out via an online phase with the cloud server $\mathcal{H}$ in order to guarantee users' privacy. In this section, we avoid such an online verification. The core idea is to randomize the certificate σ into $\sigma^\star$ such that they are unlinkable. The user can then use $\sigma^\star$ in the whole process and even the authority cannot link it with the original certificate. The Pointcheval-Sander signature perfectly matches our objective because it allows randomization and offers a highly scalable solution with signatures of constant size. Thus the authority $\mathcal{A}$ will give a signature σ on the information $\vec{I}$ of the user, which contains all necessary information such as the identity of a user, type of vaccine, effective date, etc.

For practical reason and for the desirable performance of the system, the health authority $\mathcal{A}$ only issues a long-term certificate for a user and delegates the generation of temporary tokens (which can only be used for a short time) to an untrusted cloud server $\mathcal{H}$.

It is then the question of how users can request tokens from the cloud server $\mathcal{H}$. The easiest way is for the user to show the cloud server $\mathcal{H}$ the randomized signature $\sigma^\star$ on the information $\vec{I}$. However, in this case, the cloud server $\mathcal{H}$ has all the personal information because it can get $\vec{I}$. Interestingly, the Pointcheval-Sander signature allows us to transform the signature σ on the information $\vec{I}$ into a randomized signature $\sigma^\star$ on the committed value of $\vec{I}$. In this way, the cloud server $\mathcal{H}$ can verify that the user has a valid certificate of the authority $\mathcal{A}$ without knowing any personal information. The cloud server $\mathcal{H}$ can then release the tokens to the users.

Each token contains a signature from the cloud server $\mathcal{H}$ on the committed value of $\vec{I}$ and on additional information t. The information t, appended by $\mathcal{H}$, includes essentially the redeem time for the token to avoid any reuse of the token. To verify the validation of the token, the service $\mathcal{S}$ simply checks the validity of the signature and therefore the verification is offline. The user can also randomize the received token tok to $\mathsf{tok}^\star$ and only store $\mathsf{tok}^\star$ in the memory so that even if the authority and the cloud server collude, they cannot link the used token $\mathsf{tok}^\star$ with personal information and the privacy is guaranteed in the strong sense.

Optionally, we also propose a *light verification:* when entering the cinema, restaurant, etc, it would be enough for the service to check if the token is a valid signature of the service $\mathcal{H}$. In this case, the user just needs to present their token tok to the $\mathcal{S}$ with aggregated information V on its personal information to show that the token tok is a valid signature from the cloud $\mathcal{H}$. In this case, the $\mathcal{S}$ can quickly check the validity of the token and do not get any personal information. The latter is good for privacy but its shortcoming is that the token can be transferred from one user to another as personal information is not disclosed. During an important check-in such as at the airport, at the border, the user must present his identity card. In this case, the user is required to present the information $\vec{I}$ so that the $\mathcal{S}$ can check the validity of the token on the personal information $\vec{I}$ and it needs the normal verification. In practice, we can also combine light and full verification: in all daily activities (restaurant, cinema, public transports, etc), one mainly checks light verification and occasionally and randomly check full verification to avoid the risk of transfer of tokens from one to another.

4.3.2 Construction.

Our Signature-based vaccine passport construction is presented in Figure 4. It closely follows the technical overview described above. As it relies on Pointcheval-Sander [PS16] signature, let us briefly recall the multi-message version of this signature.

Setup: A type 3 bilinear map $e : G_1 \times G_2 \to G_T$ with $G_1 =< g_1 >, G_2 =< g_2 >$, and $G_T =< g_T >$ are cyclic group of prime order p.

KeyGen: Choose a secret key $sk = (x, y_1, \ldots, y_n)$ and computes the public key $pk = (g_2, X, Y_1, \ldots, Y_n)$, where $X = g_2^x$ and $Y_i = g_2^{y_i}, i = 1, \ldots, n$.

Sign$(sk, \{m_1, \ldots, m_n\} \in (\mathbb{Z}_p^\star)^n)$: Choose a random $h \in G_1$ and define

- $\sigma_1 = h$
- $\sigma_2 = h^{x+\Sigma_{j=1}^n y_j m_j}$

and outputs $\sigma = (\sigma_1, \sigma_2)$

Verify$(sk, \sigma = (\sigma_1, \sigma_2))$**:** Check whether $\sigma_1 \neq 1_{G_1}$ (where 1_{G_1} denotes the identity in G_1) and $e(\sigma_1, X\Pi_{j=1}^n Y_j^{m_j}) = e(\sigma_2, g_2)$ are both satisfied. In the positive case, it accepts, otherwise it rejects.

We will use this signature for both authority server $\mathcal{S}$ and cloud server $\mathcal{H}$. In particular, the cloud server $\mathcal{H}$ will sign on the committed value of $\{m_1, \ldots, m_n\}$ so that users' personal information is not revealed to the cloud server $\mathcal{H}$.

4.3.3 Security.

We now analyze the security of our signature-based construction according to our desirable security and privacy of a digital vaccine passport system.

Token Unlinkability and Unforgeability. These properties are directly inherent from the unlikability and unforgeability of Pointcheval-Sanders signature as each token is exactly a signature of the cloud server $\mathcal{H}$.

Token Unreusability. for each token, the cloud server $\mathcal{H}$ appends an additional information t containing a redeem time for the token, thus the unreusability outside this redeem time comes from the unforgeability of the Pointcheval-Sanders signature.

Anonymous Identity. In this design, we achieve a high level of privacy. The users' anonymity is guaranteed against the collusion of the authority $\mathcal{S}$ and the cloud server $\mathcal{H}$. Indeed, we remark that:

- The authority $\mathcal{S}$ stores all the information $\vec{I} = \{I_1, \ldots, I_n\}$ of each user as well as the corresponding signature $\sigma = (\sigma_1, \sigma_2)$.
- The cloud server $\mathcal{H}$ stores all the users' randomized signatures $\sigma^\star = (\sigma_1^\star, \sigma_2^\star)$
- The clients only use their randomized tokens given by the cloud $\mathcal{H}$. After randomization, the users do not need to store the original signature σ and the token tok.
- The tokens used by the client $\mathsf{tok}^\star$ are unlinkable to tok and thus unlikable to $\sigma^\star$ and σ.

PARAMETERS:

- A client $\mathcal{C}$, an authority $\mathcal{A}$, a service $\mathcal{S}$, and a cloud server $\mathcal{H}$..
- A bilinear map of type 3 $e : G_1 \times G_2 \rightarrow G_T$, where $G_1 =< g_1 >, G_2 =< g_2 >$, and $G_T =< g_T >$ are cyclic groups of prime order p.
- A hash function $H : \{0,1\}^\star \rightarrow \mathbb{Z}_p^\star$
- $\mathcal{S}$ chooses a secret key $sk_{\mathcal{S}} = (x, y_1, \ldots, y_n)$ and computes the public key $pk_{\mathcal{S}} = (g_2, X, Y_1, \ldots, Y_n)$, where $X = g_2^x$ and $Y_i = g_2^{y_i}, i = 1, \ldots, n$.
- $\mathcal{H}$ chooses a secret key $sk_{\mathcal{H}} = (a, b_1, \ldots, b_n, , b_{n+1})$ and computes the public key $pk_{\mathcal{H}} = (g_2, A, B_1, \ldots, B_n, B_{n+1})$, where $A = g_2^a$ and $B_i = g_2^{b_i}, i = 1, \ldots, n+1$.

INPUTS:

- When a client $\mathcal{C}$ is vaccinated, it is associated with an information vector $\vec{I} = \{I_1, \ldots, I_n\} \in (\{0,1\}^\star)^n$, where I_1 represent the client's ID, other $I_i \in \{0,1\}^\star$ represents an information about the vaccine taken by the client such as "type of vaccine", "effective date" etc.

PROTOCOL:

I. **Registration Phase**
   - Client $\mathcal{C}$ sends $\vec{I} = \{I_1, \ldots, I_n\}$ to the authority $\mathcal{S}$ who computes $m_i = H(I_i)$ and gets the vector $\vec{m} = \{m_1, \ldots, m_n\} \in (\mathbb{Z}_p^\star)^n$.
   - Authority $\mathcal{S}$ signs on $\vec{m}$ with the Pointcheval-Sanders signature and outputs $\sigma = (\sigma_1, \sigma_2)$, where:
     * $\sigma_1 = h$, for a random $h \in G_1$
     * $\sigma_2 = h^{x + \Sigma_{j=1}^n y_j m_j}$
   - Client receives the credential $\sigma = (\sigma_1, \sigma_2)$ and checks whether $\sigma_1 \neq 1_{G_1}$ and $e(\sigma_1, X\Pi_{j=1}^n Y_j^{m_j}) = e(\sigma_2, g_2)$. If it is positive (otherwise it aborts) then $\mathcal{C}$ randomizes it by randomly chooses $r \in \mathbb{Z}_p$ and outputs $\sigma^\star = (\sigma_1^\star = \sigma_1^r, \sigma_2^\star = \sigma_2^r)$.
   - The client computes a committed value of its data: $com = (M_1, \ldots, M_n)$, where $M_j = (\sigma_1^\star)^{m_j}$, for $j = 1, \ldots, n$.

II. **Tokens Generation Phase**
   - Client $\mathcal{C}$ sends to $\mathcal{H}$ the committed value com and $\sigma^\star$.
   - Upon receipt com and $\sigma^\star$, the server checks the validity whether $\sigma_1^\star \neq 1_{G_1}$ and $e(\sigma_1^\star, X)\Pi_{j=1}^n e(M_j, Y_j) = e(\sigma_2^\star, g_2)$.
   - If it is positive (otherwise it aborts), then $\mathcal{H}$ generates tokens for the user.
     * Define $t \in (\mathbb{Z}_p^\star)$ to encode the redeem time for the token.
     * $\mathcal{H}$ randomly chooses $\lambda \in \mathbb{Z}_p^\star$ and computes $f = (\sigma_1^\star)^\lambda$ (equivalently, f is randomly generated from G_1).
     * $\mathcal{H}$ sets $\mathsf{tok} = (\mathsf{tok}_1, \mathsf{tok}_2)$ where $\mathsf{tok}_1 = f$, $\mathsf{tok}_2 = f^a(\Pi_{j=1}^n M_j^{b_j})^\lambda f^{tb_{n+1}} = f^{a + \Sigma_{j=1}^n b_j m_j + b_{n+1} t}$. Each token is thus a Pointcheval-Sanders signature of $\mathcal{H}$ on $(m_1, \ldots, m_n, t)$
   - $\mathcal{H}$ sends back (tok, t) to the client $\mathcal{C}$.

III. **Token's Redeem Phase**:
   - Upon receipt a token $\mathsf{tok} = (\mathsf{tok}_1, \mathsf{tok}_2)$ and t, the client randomizes it as $\mathsf{tok}^\star = (\mathsf{tok}_1^\star = \mathsf{tok}_1^\gamma, \mathsf{tok}_2^\star = \mathsf{tok}_2^\gamma)$, for a random $\gamma \in \mathbb{Z}_p$.
   - Verification: the client shows $\mathsf{tok}^\star$ and its information $\vec{I} = \{I_1, \ldots, I_n\}$ and t to the service who checks the validity whether: $e(\mathsf{tok}_1^\star, A(\Pi_{j=1}^n B_j^{m_j})B_{n+1}^t) = e(\mathsf{tok}_2^\star, g_2)$, where $m_i = H(I_i)$ is computed by the service.
   - Light verification: The client pre-compute $V = A(\Pi_{j=1}^n B_j^{m_j})$ and then shows $\mathsf{tok}^\star$ and V, t to the service who simply checks the validity whether: $e(\mathsf{tok}_1^\star, VB_{n+1}^t) = e(\mathsf{tok}_2^\star, g_2)$.

Figure 4: Our signature-based construction.

Again, the unlinkability comes directly from the unlinkability of the Pointcheval-Sanders signature, which is guaranteed under the DDH assumption. The privacy of the users is thus inherently preserved.

Evidently, as the user has to present its personal information to the $\mathcal{S}$, the privacy is not guaranteed if the $\mathcal{S}$ is corrupted. In this latter case, we provide the option of light verification where the service $\mathcal{S}$ only gets $V = A(\Pi_{j=1}^{n} B_j^{m_j})$ and $\mathsf{tok}^\star, t$ and cannot obtain personal information from it. Even if the service $\mathcal{S}$ colludes with the cloud $\mathcal{H}$, they also cannot get any personal information because $\mathsf{tok}^\star$ is unlikable to $(M_1, \ldots, M_n)$ (from unlikability of Pointcheval-Sanders). However, the collusion of all three parties $\mathcal{S}, \mathcal{H}, \mathcal{S}$ can still reveal the identity of the user that matches the pre-calculated value V. Indeed, the authority $\mathcal{S}$ can do an exhaustive search on the whole set of registered users with their personal information $(m_1, \ldots, m_n)$ and check if V matches $A(\Pi_{j=1}^{n} B_j^{m_j})$.

Finally, we present the security of our signature-based construction by the following theorem. Similar to Theorem 4.2.3, the security proof of the construction straightforwardly follows from the security of its building blocks and the security discussion above. Thus, we omit its proof.

Given the randomizable (Pointcheval-Sanders) signature scheme described in Section 3.5, the signature-based construction of Figure 4 securely implements the digitial vaccine passport described in Section 2 in semi-honest setting.

4.3.4 Complexity.

The use of Pointcheval-Sanders signature is particularly relevant for resource-constrained devices that have limited capacities for storage. Indeed, as the size of the signature is constant, each token contains only two elements in G_1. In term of computation:

- Client $\mathcal{C}$ computes 2 pairings and n exponentiations in G_1 for verification of the validity of each credential or each token. In practice, the $\mathcal{C}$ may directly use credential or token without the need of verifying them.

- More importantly, client $\mathcal{C}$ has to randomize credential or each token for privacy. Fortunately, it only requires 2 exponentiations in G_1 for randomization of each credential or each token.

- The cloud server $\mathcal{H}$ computes $n + 2$ pairings for verification of each credential because the client $\mathcal{C}$ only gives $\mathcal{H}$ the committed values $\mathit{com} = (M_1, \ldots, M_n)$.

- Service $\mathcal{S}$ computes 2 pairings and n exponentiations in G_1 for verification (no exponentiation is needed in light verification) of each token.

- The generation of credential or token is efficient with one exponentiation in the group G_1

5 Performance

In this section, we estimate the performance of our PrivPass system in order to show that PrivPass is feasible in practice. Assuming that a redeem token is generated every 15 minutes. For approximately 20 hours a day, then each user has $N = 80$ distinct tokens per day. If we assume that user information only consists of its identity I_1 and the concatenated vaccine information I_2, we can set $n = 2$.

For the PIR-based construction, we implement PRF and PRG instances using AES. Each AES costs 10 cycles, thus we expect to compute an AES cost $1/230 = 0.005$ microseconds using 2.3 Ghz machine. In our constructions, the participants require to compute exponentiation. For example, the DH-based private matching consists of 3 exponentiation. [PRTY19, Table 2] reports the computation cost of DH-based PSI which computes 2^{21} exponentiations in 1148.1 seconds using

		AES	**Exp.**	**Keyword-PIR**	**PIR-with-Default**	**PM**	**Pairing**
Computation (microsecond)	**Sender**	0.005	54	80000	2430	108	1620
	Receiver			7620	2430	108	
Communication/Size (KB)		0.016	0.032	480	7000	0.096	0.064

Table 1: Estimated running time and communication cost (size) for building blocks and core operations used in our PrivPass constructions.

	PIR-based Construction		Signature-based Construction	
	Runtime (ms)	Comm. (KB)	Runtime (ms)	Comm. (KB)
Authority $\mathcal{A}$	0.054	0.064	0.054	0.064
Cloud Server $\mathcal{H}$	85.24	480	226492	67108
	5.24*	7000*		
Client $\mathcal{C}$	0.004	0.016	1.836	0.352
	12.99*	0.112*	1.944*	0.384
Service $\mathcal{S}$	7.62	480	142.56	7.68
	2.54*	7000.1*	13.82*	5.12*

Table 2: Estimated running time and communication cost for our PrivPass system across different implementation options. The client generates $N = 80$ tokens per day. The cloud server has 2^{20} tokens. The numbers with "star" indicate the cost for Step (III,3) where private matching and PIR-with-default are required in the PIR-based construction. The "star" also indicates the cost of light verification in the signature-based construction.

the miracl library [2]. Using libsodium library [3] which is approximately 10× faster than miracl, we estimate that the time per exponentiation is $1148.1s/2^{21}/10 = 54$ microseconds. Our signuture-based construction requires participants compute pairings. We estimate that each pairing consists about 30× exponentiations [CDS20] which cost about 1620 microseconds.

As we mention in Section 3.3, a Keyword-PIR query on a database of size 2^{20} costs 7.62 milliseconds on the client's side and 80 milliseconds on the server's side (online time). The queries require 480KB bandwidth communication. The 2^8 PIR-with-Default queries [LPR+20] can lookup on a database of size 2^{20} with at most the communication of 7MB and the online computation of 2.43 milliseconds.

Table 1 summarizes the estimated running time and communication/size for AES, exponentiation (Exp), two PIR variants (Keyword-PIR and PIR-with-Default) with different running time on the side of sender/client and receiver/server, private matching (PM), and pairing. Note that Keyword-PIR consists of a fixed cost for an offline phase on the cloud server's side, which does not include in Table 1.

Given Table 1, we calculate the running time and communication cost for our PrivPass system across different implementation options. We report the estimated numbers in Table 2.

We observe that our PIR-based construction is faster than our signature-based in most cases. However, it requires more bandwidth cost and the (online) connection between the cloud server $\mathcal{H}$ and the service $\mathcal{S}$.

As can be seen from Table 1, a service (e.g. an airport service counter) can perform an online verification to verify whether a token is valid in only 7 ms as the service is authorized to collect the passenger's ID. For the offline verification, our protocol takes up to 0.15 seconds. Therefore, we

[2]Experiments were done on a machine with an Intel(R) Xeon(R) E5-2699 v3 2.30GHz CPU and 256 GB RAM
[3]https://doc.libsodium.org/

believe that the proposed PrivPass system is feasible in practice.

References

[ABB$^+$20] Hannah Alsdurf, Edmond Belliveau, Yoshua Bengio, Tristan Deleu, Prateek Gupta, Daphne Ippolito, Richard Janda, Max Jarvie, Tyler Kolody, Sekoul Krastev, Tegan Maharaj, Robert Obryk, Dan Pilat, Valerie Pisano, Benjamin Prud'homme, Meng Qu, Nasim Rahaman, Irina Rish, Jean-Francois Rousseau, Abhinav Sharma, Brooke Struck, Jian Tang, Martin Weiss, and Yun William Yu. Covi white paper, 2020.

[ACLS18] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with compressed queries and amortized query processing. In *2018 IEEE Symposium on Security and Privacy*, pages 962–979. IEEE Computer Society Press, May 2018.

[AIS20] Fraunhofer AISEC. Pandemic contact tracing apps: Dp-3t, pepp-pt ntk, and robert from a privacy perspective. Cryptology ePrint Archive, Report 2020/489, 2020. https://eprint.iacr.org/2020/489.

[ALP$^+$19] Asra Ali, Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Phillipp Schoppmann, Karn Seth, and Kevin Yeo. Communication–computation trade-offs in PIR. Cryptology ePrint Archive, Report 2019/1483, 2019. https://eprint.iacr.org/2019/1483.

[AMBFK16] Carlos Aguilar-Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. Xpir : Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies*, 2016(2):155 – 174, 2016.

[BBV$^+$20] Alex Berke, Michiel Bakker, Praneeth Vepakomma, Kent Larson, and Alex 'Sandy' Pentland. Assessing disease exposure risk with location data: A proposal for cryptographic preservation of privacy, 2020.

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 337–367. Springer, Heidelberg, April 2015.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1292–1303. ACM Press, October 2016.

[CBB$^+$20] Claude Castelluccia, Nataliia Bielova, Antoine Boutet, Mathieu Cunche, Cédric Lauradoux, Daniel Le Métayer, and Vincent Roca. DESIRE: A Third Way for a European Exposure Notification System Leveraging the best of centralized and decentralized systems. working paper or preprint, May 2020.

[CDF$^+$20] David Culler, Prabal Dutta, Gabe Fierro, Joseph E. Gonzalez, Nathan Pemberton, Johann Schleier-Smith, K. Shankari, Alvin Wan, and Thomas Zachariah. Covista: A unified view on privacy sensitive mobile contact tracing effort, 2020.

[CDS20] Rémi Clarisse, Sylvain Duquesne, and Olivier Sanders. Curves with fast computations in the first pairing group. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *Cryptology and Network Security - 19th International Conference, CANS 2020,* *Vienna, Austria, December 14-16, 2020, Proceedings*, volume 12579 of *Lecture Notes in Computer Science*, pages 280–298. Springer, 2020.

[CGH$^+$20] Justin Chan, Shyam Gollakota, Eric Horvitz, Joseph Jaeger, Sham Kakade, Tadayoshi Kohno, John Langford, Jonathan Larson, Sudheesh Singanamalla, Jacob Sunshine, and Stefano Tessaro. Pact: Privacy sensitive protocols and mechanisms for mobile contact tracing, 2020.

[CGN98] Benny Chor, Niv Gilboa, and Moni Naor. Private information retrieval by keywords. Cryptology ePrint Archive, Report 1998/003, 1998. https://eprint.iacr.org/1998/003.

[CK20] Henry Corrigan-Gibbs and Dmitry Kogan. Private information retrieval with sublinear online time. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 44–75. Springer, Heidelberg, May 2020.

[CKL$^+$20] Ran Canetti, Yael Tauman Kalai, Anna Lysyanskaya, Ronald L. Rivest, Adi Shamir, Emily Shen, Ari Trachtenberg, Mayank Varia, and Daniel J. Weitzner. Privacy-preserving automated exposure notification. Cryptology ePrint Archive, Report 2020/863, 2020. https://eprint.iacr.org/2020/863.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

[CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 402–414. Springer, Heidelberg, May 1999.

[DC14] Changyu Dong and Liqun Chen. A fast single server private information retrieval protocol with low communication cost. In Mirosław Kutyłowski and Jaideep Vaidya, editors, *Computer Security - ESORICS 2014*, pages 380–399, Cham, 2014. Springer International Publishing.

[DG16] Zeev Dvir and Sivakanth Gopi. 2-server pir with subpolynomial communication. *J. ACM*, 63(4), September 2016.

[DH06] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, September 2006.

[DIL$^+$20] Samuel Dittmer, Yuval Ishai, Steve Lu, Rafail Ostrovsky, Mohamed Elsabagh, Nikolaos Kiourtis, Brian Schulte, and Angelos Stavrou. Function secret sharing for psi-ca: With applications to private contact tracing. Cryptology ePrint Archive, Report 2020/1599, 2020. https://eprint.iacr.org/2020/1599.

[DPT20] Thai Duong, Duong Hieu Phan, and Ni Trieu. Catalic: Delegated PSI cardinality with applications to contact tracing. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 870–899. Springer, Heidelberg, December 2020.

[DRRT18] Daniel Demmler, Peter Rindal, Mike Rosulek, and Ni Trieu. Pir-psi: Scaling private contact discovery. *Proceedings on Privacy Enhancing Technologies*, 2018(4), 2018.

[EUC] European digital green certificates. https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/covid-19-digital-green-certificates_en.

[FNP04] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 1–19. Springer, Heidelberg, May 2004.

[FNW96] Ronald Fagin, Moni Naor, and Peter Winkler. Comparing information without leaking it. *Commun. ACM*, 39(5):77–85, May 1996.

[Goo19] Google. https://blog.google/technology/safety-security/password-checkup. 2019.

[Goo20] Apple and google privacy-preserving contact tracing. https://www.apple.com/covid19/contacttracing, 2020.

[GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 803–815. Springer, Heidelberg, July 2005.

[IPS] IPSOS. Global public backs covid-19 vaccine passports for international travel. https://www.ipsos.com/en/global-public-backs-covid-19-vaccine-passports-international-travel.

[KKRT16] Vladimir Kolesnikov, Ranjit Kumaresan, Mike Rosulek, and Ni Trieu. Efficient batched oblivious PRF with applications to private set intersection. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 818–829. ACM Press, October 2016.

[KMP$^+$17] Vladimir Kolesnikov, Naor Matania, Benny Pinkas, Mike Rosulek, and Ni Trieu. Practical multi-party private set intersection from symmetric-key techniques. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1257–1272. ACM Press, October / November 2017.

[KMR12] Seny Kamara, Payman Mohassel, and Ben Riva. Salus: A system for server-aided secure function evaluation. Cryptology ePrint Archive, Report 2012/542, 2012. https://eprint.iacr.org/2012/542.

[KRT18] Vladimir Kolesnikov, Mike Rosulek, and Ni Trieu. Swim: Secure wildcard pattern matching from ot extension. In Sarah Meiklejohn and Kazue Sako, editors, *Financial Cryptography and Data Security - 22nd International Conference, FC 2018, Revised Selected Papers*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 222–240. Springer Verlag, 2018. Funding Information: he first author was supported by Office of Naval Research (ONR) contract number N00014-14-C-0113. The second and third authors were supported by NSF awards 1149647 and 1617197.; 22nd International Conference on Financial Cryptography and Data Security, 2018 ; Conference date: 26-02-2018 Through 02-03-2018.

[LAY$^+$20] Joseph K. Liu, Man Ho Au, Tsz Hon Yuen, Cong Zuo, Jiawei Wang, Amin Sakzad, Xiapu Luo, and Li Li. Privacy-preserving covid-19 contact tracing app: A zero-knowledge proof approach. Cryptology ePrint Archive, Report 2020/528, 2020. https://eprint.iacr.org/2020/528.

[LPR$^+$20] Tancrède Lepoint, Sarvar Patel, Mariana Raykova, Karn Seth, and Ni Trieu. Private join and compute from pir with default. Cryptology ePrint Archive, Report 2020/1011, 2020. https://eprint.iacr.org/2020/1011.

[LTKS20] Xiaoyuan Liu, Ni Trieu, Evgenios M. Kornaropoulos, and Dawn Song. Beetrace: A unified platform for secure contact tracing that breaks data silos. *IEEE Data Eng. Bull.*, 43(2):108–120, 2020.

[Mea86] Catherine A. Meadows. A more efficient cryptographic matchmaking protocol for use in the absence of a continuously available third party. In *IEEE Symposium on Security and Privacy*, pages 134–137, 1986.

[Mer90] Ralph C. Merkle. A certified digital signature. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 218–238. Springer, Heidelberg, August 1990.

[MMRV20] Parthasarathy Madhusudan, Peihan Miao, Ling Ren, and V.N. Venkatakrishnan. Contrail: Privacy-preserving secure contact tracing. https://github.com/ConTraILProtocols/documents/blob/master/ContrailWhitePaper.pdf, 2020.

[Ode09] Goldreich Oded. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, USA, 1st edition, 2009.

[PRTY19] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. SpOT-light: Lightweight private set intersection from sparse OT extension. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 401–431. Springer, Heidelberg, August 2019.

[PRTY20] Benny Pinkas, Mike Rosulek, Ni Trieu, and Avishay Yanai. PSI from PaXoS: Fast, malicious private set intersection. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, volume 12106 of *LNCS*, pages 739–767. Springer, Heidelberg, May 2020.

[PS16] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, February / March 2016.

[PS18] David Pointcheval and Olivier Sanders. Reassessing security of randomizable signatures. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 319–338. Springer, Heidelberg, April 2018.

[PSTY19] Benny Pinkas, Thomas Schneider, Oleksandr Tkachenko, and Avishay Yanai. Efficient circuit-based PSI with linear communication. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 122–153. Springer, Heidelberg, May 2019.

[RPB20] Ramesh Raskar, Deepti Pahwa, and Robson Beaudry. Contact tracing: Holistic solution beyond bluetooth. *IEEE Data Eng. Bull.*, 43(2):67–70, 2020.

[Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.

[TPH$^+$20] Carmela Troncoso, Mathias Payer, Jean-Pierre Hubaux, Marcel Salathé, James Larus, Edouard Bugnion, Wouter Lueks, Theresa Stadler, Apostolos Pyrgelis, Daniele Antonioli, Ludovic Barman, Sylvain Chatel, Kenneth Paterson, Srdjan Čapkun, David Basin, Jan Beutel, Dennis Jackson, Marc Roeschlin, Patrick Leu, Bart Preneel, Nigel Smart, Aysajan Abidin, Seda Gürses, Michael Veale, Cas Cremers, Michael Backes, Nils Ole Tippenhauer, Reuben Binns, Ciro Cattuto, Alain Barrat, Dario Fiore, Manuel Barbosa, Rui Oliveira, and José Pereira. Decentralized privacy-preserving proximity tracing, 2020.

[TSS$^+$20] Ni Trieu, Kareem Shehata, Prateek Saxena, Reza Shokri, and Dawn Song. Epione: Lightweight contact tracing with strong privacy. *CoRR*, abs/2004.13293, 2020.

[TZBS20] Amee Trivedi, Camellia Zakaria, Rajesh Balan, and Prashant Shenoy. Wifitrace: Network-based contact tracing for infectious diseases using passive wifi sensing, 2020.

[vABB$^+$20] Sydney von Arx, Isaiah Becker-Mayer, Daniel Blank, Jesse Colligan, Rhys Fenwick, Mike Hittle, Mark Ingle, Oliver Nash, Victoria Nguyen, James Petrie, Jeff Schwaber, Zsombor Szabo, Akhil Veeraghanta, Mikhail Voloshin, Tina White, and Helen Xue. Slowing the spread of infectious diseases using crowdsourced data. *IEEE Data Eng. Bull.*, 43(2):71–82, 2020.