Enhancing the Accountability of Single Sign-On Services with Privacy-Preserving Public Logs

Guangqi Liu, Qiongxiao Wang, Jingqiang Lin, Dawei Chu, Wei Wang, Xiaokun Zhang, Fengjun Li

***Abstract*—Single sign-on (SSO) is becoming more and more popular on the Internet. An SSO ticket issued by the identity provider (IdP) allows an entity to sign onto a relying party (RP) on behalf of the identity enclosed in the ticket. In order to ensure its authenticity, an SSO ticket is signed by the IdP and verified by the RP. However, security incidents indicate that a signing system (e.g., certification authority) might be compromised to sign fraudulent messages, even when it is protected in accredited systems. Compared with certification authorities, the online signing components of IdPs are even more exposed to adversaries and thus more vulnerable to such threats in practice; for example, the recent SolarWinds incident happened with the first recorded use of Golden SAML attacks to compromise an IdP system. This paper proposes *ticket transparency* to enhance the accountability of SSO services with privacy-preserving public logs, against fraudulent tickets issued by a potentially compromised IdP. With this accountable scheme, an IdP-signed ticket is accepted by the RP only if it is also recorded in the public logs. It enables a user to check all his tickets in the logs and detect any fraudulent ticket issued without his participation or authorization. Ticket transparency integrates *blind signatures*, *identity-based encryption*, *Bloom filters* and *deterministic public-key encryption* in the design, to balance *transparency*, *privacy* and *efficiency* in security-enhanced SSO services. To the best of our knowledge, this is among the first attempts to solve the security problems caused by potentially intruded or compromised IdPs in the SSO services. We implemented the prototype system, and the experimental evaluation shows that ticket transparency introduces acceptable overheads in the sign-on process.**

***Index Terms*—Accountability, Privacy, Single Sign-On, Transparency, Trust**

I. Introduction

Single sign-on (SSO) services have recently become the popular identity management and authentication infrastructures in the Internet. Famous IT service providers such as Google, PayPal, Facebook and Microsoft provide their own SSO services, which allow a user to sign onto millions of networked applications but with the same account using SSO protocols such as OpenID Connect [2] and SAML with WS-Security [3] in SOAP [4].

A critical data item in the SSO services is the SSO *ticket* (e.g., id-token in OpenID Connect or assertion in SAML), which includes the authenticated user's identity, its validity period, the target relying party (RP), the nonce against replay attacks, and other optional fields. An SSO ticket is signed by the *identity provider* (IdP), after it authenticates a user. Then, the ticket is presented to the target RP. Trusting the IdP, the visited RP accepts the ticket after verifying the IdP's signature and allows the user to sign on as the identity enclosed in the ticket. So the IdP is a *trusted third party* that signs messages to provide security guarantees of user identity and authentication for the RPs. In this sense, IdPs take a similar role as certification authorities (CAs) in the public key infrastructure (PKI), which sign certificates to provide security guarantees (such as authentication, confidentiality, data integrity and non-repudiation) for various PKI-based applications [5].

However, several real-world incidents indicate that third-party security providers cannot be fully trusted as expected, because even a signing system that is protected with defense-in-depth protections could be compromised or deceived to sign fraudulent messages. For example, several well-known professional CAs, which are accredited to provide certificate services, were intruded or deceived to sign fraudulent TLS server certificates [6]–[9], which contained well-formatted but incorrect or fake information. These fraudulent certificates are then exploited to launch man-in-the-middle (MitM) attacks to intercept the private data of victims.

Compared with the signing component of CAs, which is usually built on hardware security modules (HSMs) in isolated networks, the *online* signing component of IdPs is even more exposed to adversaries and thus more vulnerable to such threats in practice. For example, in the Golden SAML attack [10], the attackers only need an unprivileged account of Microsoft Active Directory Federation Services to access the private key to sign (fraudulent) SSO tickets. The recent SolarWinds incident resulted in the first recorded use of Golden SAML attacks [11], [12]. The CVE-2021-26715 vulnerability [13] allows unauthenticated attackers to exploit the vulnerable IdP server to access any host in the internal network (e.g., an HSM to sign tickets) and obtain its response. Or, after deceiving the victim user to click a crafted link to poison the user's session, an attacker could bypass the authorization check and receive a ticket from the MITREid Connect IdP due to CVE-2021-27582 [14], [15].

A fraudulent SSO ticket allows an attacker to sign onto online applications on behalf of the victim user, which makes

Guangqi Liu, Qiongxiao Wang and Wei Wang are with State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences (Beijing 100093), and also School of Cyber Security, University of Chinese Academy of Sciences (Beijing 100049); Jingqiang Lin (corresponding author) is with School of Cyber Security, University of Science and Technology of China (Hefei 230026); Dawei Chu is with the Headquarter of Chinese Academy of Sciences (Beijing 100864); Xiaokun Zhang is with Beijing Institute, University of Science and Technology of China (Beijing 100193); and Fengjun Li is affiliated with Department of Electrical Engineering and Computer Science, the University of Kansas (KS 66045). E-mail: liuguangqi@iie.ac.cn, wangqiongxiao@iie.ac.cn, linjq@ustc.edu.cn, dwchu@cashq.ac.cn, wangwei@iie.ac.cn, xkzhang@ustc-bj.cn, fli@ku.edu.

A preliminary version of this manuscript appeared under the title "Ticket transparency: Accountable single sign-on with privacy-preserving public logs" in Proc. 15th International Conference on Security and Privacy in Communication Networks (SecureComm 2019) [1].

the IdPs be an attack target of interest [11], [12] [10], [16], [17]. Moreover, fraudulent SSO tickets are much more difficult to detect than fraudulent certificates, because an SSO ticket is presented only to a particular RP, valid for a very short period (e.g., 3 to 5 minutes typically), and transmitted ephemerally over private channels such as HTTPS. As we learned from the SolarWinds incident [11], [18], detecting such attacks is extremely difficult and needs evidence from multiple sources.

It is of great importance to build security-enhanced mechanisms to mitigate the risks by malicious or compromised third parties that were completely trusted in the past. Certificate transparency [19] is widely adopted to enhance the trustworthiness of the TLS certificate ecosystem [20]–[22]. It employs log servers to sign the CA-signed TLS server certificates again and record them in publicly-visible logs. The extra signature by log servers, called *signed certificate timestamp* (SCT), is sent along with the server certificate to browsers in TLS negotiations. Here, the SCT is a promise to make a certificate visible in public logs. Then, interested parties, especially the TLS servers whose identities are bound in (fraudulent) certificates, are enabled to search for all relevant certificates in the public logs and check whether they are issued with necessary authorization.

Certificate transparency provides a framework to monitor CAs' certificate signing operations and thus introduces transparency in the creation of CA-signed certificates. Following this idea, we extend the *public accountability* from the CAs in PKIs to the IdPs in SSO services, and propose *ticket transparency* to address the risk of fraudulent SSO tickets. Fraudulent SSO tickets are well-formatted and verifiable tickets issued by malicious or compromised IdPs but without the authorization or participation of ticket holders whose identities are in the tickets. Similar to certificate transparency, ticket transparency introduces log servers to monitor the ticket-signing operations of the IdPs. In particular, an IdP-signed ticket needs to be signed by a log server again, before it is accepted by the target RP. This extra signature by the log server is also a promise to record the SSO tickets in publicly-visible logs. Ticket transparency enables a user, who suspects some attacker signing onto any RP on behalf of him with a fraudulent ticket, to search for all tickets with his account in the logs and detect the fraudulent ones among them.

However, the certificate transparency framework cannot be directly adopted in SSO services due to privacy concerns. Different from PKI certificates, SSO tickets contain privacy information about the service requester (i.e., the user), the service provider (i.e., the RP), the occurrence time of sign-on activities, etc. To protect user privacy while enabling efficient ticket search in the accountable SSO services, we integrate blind signatures [23], identity-based encryption (IBE) [24], Bloom filters [25], and deterministic public-key encryption [26] in the design of ticket transparency. First, an SSO ticket is *blindly signed by the log server*, so the ticket content is protected against the log server. That is, the ticket is blinded before being recorded in public logs. Secondly, the secret blinding factor is logged along with the ticket in public logs, but *encrypted using the user's IBE public key* by the IdP. Therefore, only the user is able to decrypt the blinding factor, and then un-blind his tickets. This encryption is identity-based, to reduce the overhead of key management. All users' IBE private keys are generated by a *trusted coordinator*. Thirdly, *deterministic public-key encryption is simultaneously adopted to encrypt the blinding factor*, so that it can be verified by the target RP. This encryption ensures the decryption of blinding factors by the coordinator in necessary cases; for example, when a user wants to decrypt a suspected ticket but fails. Such failures might be caused by some malicious operations of the IdP. The trusted coordinator is also responsible for the dispute resolution when a user cannot decrypt suspected tickets.

Meanwhile, when the SSO protocols are deployed, *pairwise pseudonymous identifiers* (PPIDs) [2], [27] are usually adopted to protect user privacy against curious RPs. That is, when a user is signing onto different RPs, the IdP assigns a unique PPID (but not the account in the IdP) for each RP in SSO tickets, and then curious RPs cannot profile the user's network activities by linking the identities in the tickets across multiple collusive RPs. However, if the PPIDs are random strings, which is common in existing SSO systems, a user has to remember all his PPIDs to search for his tickets in the public logs. Thus, *we generate PPIDs also by deterministic public-key encryption*. We concatenate the user's identity, the RP's identity and the hash value of the user's IBE private key, as the input of deterministic public-key encryption, and the result works as PPIDs. So the user is able to re-generate all his PPIDs by himself and the trusted coordinator holding the private key is able to analyze these PPIDs in the dispute resolution.

Finally, a more efficient search of tickets is designed with Bloom filters, at the expense of limited user privacy; otherwise, a user has to try each blindly-signed ticket in the logs one by one (i.e., attempt to decrypt the blinding factor for each ticket entry), when searching for his tickets. In particular, the alias of a user's PPID, i.e., the result of *the PPID through a Bloom filter*, is stored along with each of his ticket entries in the logs, while other elements of the ticket entry are still kept encrypted or blinded. So the user is able to quickly filter out most of other users' tickets, while the occurrence time of his sign-on activities (and nothing about the visited RPs) is visible to other users. Note that this privacy leakage is controlled by the false positive rate of the Bloom filter.

Contribution. We analyze the accountability of IdPs in the SSO services. Then, ticket transparency is proposed to monitor the ticket-signing operations of the IdPs, ensuring that a fraudulent ticket issued by IdPs will be finally detected. The design of ticket transparency balances transparency, privacy and efficiency in the security-enhanced SSO services. To the best of our knowledge, *this is among the first attempts to solve the security problems caused by potentially compromised IdPs in the SSO services*. We implemented the prototype system with MITREid Connect and Connect2id Nimbus, and the experimental evaluation demonstrates that ticket transparency introduces acceptable overheads in the sign-on process.

The remainder is organized as follows. Section II introduces the background and related work. Ticket transparency is presented in Section III. Then, its security and performance are analyzed in Sections IV and V, respectively. Section VI concludes this work.

II. Background and Related Work

A. Security and Privacy of SSO Services

The security of SSO services has been investigated for a long time. Implementation vulnerabilities have been discovered in several systems, including OpenID and customized SSO protocols [28], SAML with WS-Security [29], OAuth [30]–[32], etc., which allow an attacker to sign onto RPs on behalf of other users or disclose private information. Such vulnerabilities are found in popular SSO services such as Facebook OAuth [28], [33], Windows LiveID [34] and Google OpenID [28], [35]. [17] exploits the discovery phase of OpenID to introduce a malicious IdP to an RP, and all accounts on the RP are then compromised. [16] investigates the threats magnified by SSO services, when the session between a user and the IdP is hijacked. [36] analyzes the threats and their impacts in OpenID Connect, helping to understand and mitigate some identified threats. Distributed protocols [37] keep secure SSO services as long as not all servers are corrupt at the same time. SGX-Cube [38] runs the SSO authentication task inside Intel SGX enclaves, to protect the credentials even if the host is compromised. These works focused on the secure implementation and deployment of SSO services, and they do not focus the problem caused by the fraudulent tickets that are issued by compromised IdPs. On the contrary, our work proposes a solution for the accountability of SSO services against the potentially-compromised IdPs.

SSO services introduce two threats of user privacy leakage [27]: curious RPs might track a user's network activities by linking the identities in the SSO tickets across collusive RPs, and the IdP is enabled to profile any user's network activities when issuing SSO tickets. PPIDs [2], [27] are usually adopted to protect user privacy against curious RPs; that is, given a certain user, the IdP assigns a unique PPID for each RP in SSO tickets, so that the RPs cannot link the identities across RPs. BrowserID [39] and SPRESSO [40] provide privacy-preserving SSO services, where the curious IdP is unaware of the visited RP when issuing an SSO ticket.

On the other hand, anonymous SSO schemes are proposed to allow users to access RPs without revealing their identities to the RPs, for the global system for mobile communication (GSM) [41], based on broadcast encryption [42], group signatures [43] or extended Chebyshev Chaotic Maps [44]. [45] and [46] improve the anonymous SSO schemes, where only the designated RP service is able to verify the tickets, and no identity information is released to the IdP. In summary, these privacy-preserving approaches assume honest IdPs, and then cannot handle the problem of fraudulent tickets.

We extend the preliminary version [1] as follows. This extended scheme supports PPIDs for better user privacy, which is commonly supported in existing SSO services, and we utilize deterministic public-key encryption to generate PPIDs to work with ticket transparency compatibly. Deterministic public-key encryption also works as an auxiliary of IBE, to enable the coordinator to decrypt messages without exhaustively enumerating user identities, especially for an IBE scheme with anonymous ciphertext indistinguishability against key generation center (ACI-KGC) [47], which is not considered in the preliminary scheme. Moreover, the prototype system is implemented, while the preliminary version only estimated the overheads by measuring the main cryptographic computations.

Finally, [48] discusses different strategies against fraudulent tickets in the SSO scenarios, including (*a*) ticket transparency in this paper and (*b*) ticket synchronization, another solution inspired by the designs against fraudulent certificates.

B. CA Security Incident and Certificate Transparency

Well-known accredited CAs are reported to sign fraudulent certificates due to network intrusions [6]–[9], reckless identity validations [49]–[51], misoperations [52]–[54], or government compulsions [55], [56]. Lessons are learned in an unexpected way that a signing system which is built on HSMs in well-protected organizations, could still be compromised to sign fraudulent messages, with well-formatted and verifiable but incorrect or misleading data.

Certificate transparency [19] is proposed to enhance the accountability of CAs' certificate-signing operations. After signing a certificate, the CA submits it to a log server, and the log server responds with an SCT, which is a promise to record the certificate in publicly-visible logs. A web server presents the SCTs along with its certificate in TLS negotiations; otherwise, browsers reject the server certificate. Then, a web server may periodically search for the certificates binding its identity (i.e., domain name). Once a fraudulent certificate is detected (i.e., the key pair is not held by the web server), it will take (out-of-band) actions to stop or mitigate the damage. Meanwhile, auditors periodically verify the consistency of logs to ensure that they are append-only, i.e., a (fraudulent) certificate will never be deleted or modified after being added. A browser also acts as an auditor, to check whether the certificate in the received SCT is publicly visible in the logs. [57] defined the security properties of logging schemes, and proved certificate transparency achieves these properties. [20]–[22], [58], [59], [60] investigated the wide deployment of certificate transparency from different aspects.

Certificate transparency has been extended in different ways. Revocation transparency [61] provides publicly-accountable certificate revocation. CIRT [62] achieved both certificate transparency and revocation transparency by recording certificates in two Merkle hash trees – one of which is in chronological order with all certificates and the other is lexicographical with only the recent ones for each certificate subject. Singh et al. [63] improved CIRT by bilinear-map accumulators and binary trees, to achieve transparency with shorter proofs. [64] utilized blockchain as append-only logs to record certificates and certificate revocation lists, to achieve both certificate transparency and revocation transparency.

Based on certificate transparency, binary transparency ensures Firefox binaries are publicly logged [65]. A Merkle hash tree is computed over all binaries of each version, and the root node is bound in a certificate that is transparently logged. Sigstore [66] combines certificate transparency and short-lived certificates, to provide transparent software signing services. Attestation transparency [67] combines certificate transparency and Intel SGX: a legacy client verifies the service by establishing a TLS channel, while the service private key is protected

in SGX enclaves and the server certificate is transparently logged. Inspired by the design of certificate transparency, the general transparency overlay [68] is proposed, which can be instantiated to implement transparency for other services.

Certificate transparency and its variations provide references to solve the problem of fraudulent SSO tickets, but these solutions cannot be directly adopted due to privacy concerns. Customized privacy-preserving techniques are necessary to be integrated into the SSO services and other applications. CONIKS [69] presented transparent key directories based on Merkle prefix trees, allowing its users to audit their public keys while preserving user privacy. [70] proposed privacy-preserving logging for distributed data processing, by blinding a user's public key. Insynd [71] proposed privacy-preserving transparent logging by authenticated data structures, so users take actions without disclosing everything in the logs.

CASTLE [72] attempts to eliminate fraudulent certificates, by an air-gapped and completely-touchless signing system for CAs, but it is designed for only low-volume workloads and then does not work for SSO services. Threshold signature schemes [73], [74] are applied in CAs to protect the private key to sign certificates. Distributing the private key magnifies the difficulty of compromising the key [75], [76], but multiple administrators of CASTLE are required to manually check whether the to-be-signed request includes correct information or not [72], which is impractical in online services.

C. Accountability of Third-Party Services

The public accountability of cloud services is one of the important topics of cloud security. Third-party auditors are introduced to check the integrity of outsourced data in untrusted cloud systems, while the data are not disclosed to the auditors [77], [78], [79]. Liu et al. propose consistency as a service [80]: a data cloud is maintained by the cloud service provider, and a group of users that constitute an auditor verify whether the data cloud provides the promised level of consistency. PDP [81] and POR [82] enable the tenants to verify whether the data are intact in untrusted clouds, with lightweight complexity of communications and computations. [83] detects CPU cheating on virtual machines maintained by semi-trusted cloud providers, using CPU-intensive computations. [84] explains how transparency can be deployed to achieve accountability and improve the normative fidelity of systems. The SSO services can be viewed as another kind of cloud service – ID as a service, but the accountability of untrusted IdPs has not been well investigated in the literature.

The private key generator (PKG) of IBE generates private keys for all users, so that it has to be fully trusted. This problem of inherent key escrow is mitigated by different ways: (*a*) distributed PKGs [24], [85], where the IBE master key is distributed among several independent components, (*b*) accountable IBE [86], [87] – if the PKG re-generates the private key for any user, a proof will be produced automatically, or (*c*) KGC-anonymous IBE [47] – the PKG generates the private key for an authenticated user without knowing the list of user identities. We share the same spirit with these solutions that the assumed trust of a third party needs to be reduced.

III. Accountable Single Sign-On with Privacy-Preserving Public Logs

In this section, we first present the threat model, security goals, and cryptographic building blocks. Then, the designs of ticket transparency are proposed by steps, following the requirements of transparency, privacy and efficiency.

A. Threat Model and Security Goals

Threat Model. The basic SSO scenario includes an IdP, multiple RPs, and a number of users. We assume *the RPs are honest-but-curious*. Since the RPs are the service providers who are responsible for verifying the validity of SSO tickets, we have to assume honest RPs. Ticket transparency aims to prevent unauthorized sign-on to RPs by fraudulent tickets, so the RPs are assumed to be honest in a user's SSO activities; otherwise, a dishonest RP would allow an attacker to sign on even without any valid tickets. The honest RPs strictly follow the specifications of ticket transparency. Meanwhile, RPs are curious about the users' privacy [27]. That is, in addition to the required actions of ticket transparency, RPs may take any behaviors that do not violate the specifications to infer user privacy; for example, curious RPs collude to link the identities in SSO tickets across these RPs, to track a user's activities.

The online IdP is potentially malicious and could be compromised by a number of attacks. The malicious IdP might pretend to be benign and act as expected to avoid being detected, but it could arbitrarily deviate from its specifications at times. For example, the IdP signs fraudulent tickets and then tries its best to conceal their existence by manipulating other messages.

Ticket transparency requires an independent log server to publicly record SSO tickets. *The log server is assumed to be semi-trusted.* It does not actively deviate from its specifications but might be lazy and forget recording a ticket in the logs. The log server is also curious about the users' privacy, so it tries to infer sensitive or private information about a user from the communications and the logs.

Finally, *an offline trusted coordinator* is introduced in ticket transparency. The coordinator is not involved in the online sign-on activities. The coordinator holding the IBE master key, acts as the PKG of IBE to generate private keys for all users. Since this key generation happens only once for each user, we assume it is finished in a secure offline way. The trusted coordinator also holds the system-wide private key of deterministic public-key encryption. These keys enable it to coordinate the dispute resolution process, when a user suspects there exists some fraudulent ticket labeled with his identity (or PPID) in the public logs but he cannot decrypt this ticket entry.

Security Goals. Ticket transparency cannot eliminate fraudulent tickets issued by the (compromised) IdP, but it ensures that any accepted fraudulent ticket will be detected by the victim user, the RPs and/or the coordinator in the future. In particular, if the IdP issues a ticket without the user's authorization or participation, our scheme ensures: (*a*) the fraudulent ticket without a valid signature by the log server will not be accepted by the RPs; or (*b*) the fraudulent ticket with a valid signature by the log server, accepted by the target RP, will be verified by

the RP first, and then identified from the public logs by the victim user, by comparing the ticket entries and his history of sign-on activities, sometimes with the coordinator's help. Here, we assume, when the victim user suspects that some attacker signs on using his account, he remembers all his sign-on operations, especially in the period during which the suspected sign-on action occurred.

The fraudulent-ticket detection requires the log server to record all tickets in the public logs. Therefore, our scheme also needs to audit the operations of lazy log servers, to ensure that (*a*) the log server records all valid tickets in the public logs, and (*b*) the logs are append-only. Any log operation deviating from these specifications, will be detected by the honest RPs which accept SSO tickets. Anyway, these audits against the log server may be finished by extra redundant independent auditors, as the auditors of certificate transparency [19].

The second goal is to protect user privacy, as much as possible. A user's privacy is defined as the information about his sign-on activities, such as the account (or PPID), the requested RP, the occurrence time, etc. These privacy data are included in the SSO tickets. First of all, PPIDs (but not the user account registered in the IdP) are enclosed in the SSO tickets [2], [27]. More importantly, in order to protect user privacy in the transparent SSO services, we employ blind signatures to hide the contents of SSO tickets in the public logs. However, to check whether a suspected ticket is indeed fraudulent or not, the coordinator needs to recover (i.e., unblind or decrypt) some relevant ticket entries in the dispute resolution. This process may disclose the private information of some other users. Overall, ticket transparency ensures the fraudulent-ticket detection at a cost of user privacy – compared to the original SSO protocols [2], [3], ticket transparency leaks limited user privacy to the trusted coordinator. Besides, if a Bloom filter is adopted to enable a user to search for tickets more efficiently, some privacy is leaked with a false positive rate – the occurrence time of his sign-on activities (and nothing about the visited RPs) is visible to other users.

B. Cryptographic Building Blocks

Ticket transparency is designed on top of these cryptographic building blocks: *blind signature*, *identity-based encryption*, *Bloom filters* and *deterministic public-key encryption*. We define the notations and explain each of these building blocks as follows.

- I, L, C, and R_i, denote the IdP, the log server, the coordinator, and the i-th RP, respectively.
- A is the authenticated user, i.e., the victim of the malicious operations by the IdP. Z is another user of the SSO system, malicious sometimes.

Blind Signature. With a secret factor s, I blinds (or encrypts) a message m into $m' = \mathbb{B}(m, s)$, and L blindly signs m' into $\mathbb{S}(m') = \{m', Sig_{L,m'}\}$. Then, I un-blinds the signature signed by L to obtain $Sig_{L,m} = \mathbb{U}(Sig_{L,m'}, s)$, where $Sig_{L,m}$ is the "plain" signature of m by L and can be verified using L's public key.

Identity-based Encryption (IBE). The trusted coordinator holds the IBE master key and generates the private keys for all users. The IBE public parameters are publicly known, and everyone is able to derive A's IBE public key and encrypt s into $\mathbb{E}^{\mathcal{I}}_A(s)$, but only A himself or the coordinator can decrypt $\mathbb{E}^{\mathcal{I}}_A(s)$ into s. The superscript $\mathcal{I}$ indicates the operations of IBE, to distinguish with that of deterministic public-key encryption in the remainder, where the superscript $\mathcal{D}$ is used.

Bloom Filter. We adopt a Bloom filter $\mathbb{F}(\cdot)$ to generate an alias for A in the form of $N_{A,R_i} = \mathbb{F}(PID_{A,R_i})$, where PID_{A,R_i} is the PPID of A for R_i. $\mathbb{F}(PID_{A,R_i})$ is a vector of compact representations of PID_{A,R_i}, and these representations are concatenated as the alias. The Bloom filter is deterministic, with a false negative rate of zero and a false positive rate of α, where $0 < \alpha < 1$. That is, the expectation of the probability $P(X \neq PID_{A,R_i} | \mathbb{F}(X) = \mathbb{F}(PID_{A,R_i}))$ is α.

Deterministic Public-Key Encryption. We adopt deterministic public-key encryption to (*a*) generate the PPIDs and (*b*) verify the correct encryption of blinding factors, while only the trusted coordinator holds the private key. A deterministic public-key encryption algorithm encrypts messages without randomness; that is, anyone encrypts m, and the result $\mathbb{E}^{\mathcal{D}}(m)$ is unique. Note that deterministic encryption cannot protect messages when the plaintext space is small [26].

C. The Ticket Transparency Framework by Steps

We elaborate the design of ticket transparency step by step, following the requirements of *transparency*, *privacy* and *efficiency* of the accountable SSO services.

1. Transparent Ticket. In the original SSO protocols, when attempting to sign onto R_i, A is redirected from R_i to I, who authenticates A and then signs a ticket $m = \{A, R_i, o\}$ into $\mathbb{S}_I(m) = \{m, Sig_{I,m}\}$, where o denotes the other necessary information in the ticket specified in the SSO protocols. Then, I sends the signed ticket to complete the sign-on process.

Improved Design. Instead of sending $\mathbb{S}_I(m)$ to A, I sends it to L, which signs the ticket in the form of $\mathbb{S}_L(m)$ and returns it to I. L also records this *transparent ticket* $\{m, Sig_{I,m}, Sig_{L,m}\}$ in public logs. On receiving $\mathbb{S}_L(m)$, I forwards $\{m, Sig_{I,m}, Sig_{L,m}\}$ to A, who presents the transparent ticket to R_i to complete the sign-on process.

Ticket Verification. After R_i verifies two signatures on the transparent ticket (and conducts other verifications as specified in the original SSO protocols, including the validity period, the unique identifier against replay attacks, etc.), it accepts the ticket and allows the ticket holder to sign on as A.

Transparency. This is a straightforward extension of certificate transparency to the SSO scenario. It creates a set of transparent tickets and records them in the public logs. The IdP (or the signing system of an IdP) can no longer issue a valid ticket only by itself, as the ticket has to be signed again by the log server. The tickets in the public logs are always visible to enable the fraudulent-ticket detection at any time. Whenever any user suspects that an attacker is signing onto R_i on behalf of him exploiting a fraudulent ticket, he will retrieve all tickets labeled with his account from the public logs and detect fraudulent tickets among them.

2. Blindly-Signed Transparent Ticket. While the basic design of ticket transparency defends against compromised IdPs

effectively, it leaks user privacy because all sign-on activities are recorded in the public logs. So we revise the design to *blindly-signed transparent tickets*, where the second signatures are created blindly by the log server, to eliminate the privacy information in publicly-visible ticket entries.

Improved Design. After signing a ticket, I blinds it into $m' = \mathbb{B}(m, s)$ with a secret random blinding factor s. I sends m', instead of m, to L along with $Sig_{I,m}$ and $\mathbb{E}_A(s)$, which enables A to un-blind the ticket later. Accordingly, L blindly signs m' into $\mathbb{S}_L(m')$ and returns it to I. I un-blinds $\mathbb{S}_L(m')$ into $\mathbb{S}_L(m)$, and sends $\{m, Sig_{I,m}, Sig_{L,m}\}$ to A. In the meantime, L creates a ticket entry $\{\mathbb{S}_L(\mathbb{B}(m, s)), Sig_{I,m}, \mathbb{E}_A(s)\}$ and records it chronologically in the public logs.

Privacy and Transparency. There is no plaintext information logged in the logs, which protects user privacy from irrelevant entities (i.e., the log server and any other entities without the key to decrypt s). When a user suspects he is the victim of any fraudulent tickets, he retrieves all ticket entries within the suspected period and applies his private key to decrypt $\mathbb{E}_A(s)$ of every ticket entry. For the tickets issued under his account, he is enabled to recover s correctly, and then un-blind and verify the signatures. Thus, any user can only recover his own transparent tickets, but not the ones under others' accounts.

3. Encrypted Blinding Factor. The above scheme works, if the compromised IdP is not intelligent or collusive with malicious users. However, the compromised IdP could destroy the fraudulent-ticket detection by issuing fraudulent tickets under A's account but blinding them with a blinding factor escrowed to another user Z, who is malicious or even does not exist. More specifically, a malicious IdP blinds a ticket as $m' = \mathbb{B}(m, s)$ and sends it along with $\mathbb{E}_Z(s)$ to the log server. This results in a ticket entry $\{\mathbb{S}_L(\mathbb{B}(m, s)), Sig_{I,m}, \mathbb{E}_Z(s)\}$ in the logs. When A retrieves this entry, he cannot decrypt the blinding factor to check whether it is a fraudulent ticket under his account or not.

We then adopt IBE, utilizing the inherent key escrow of IBE to ensure that the blinding factor can always be decrypted when necessary. The IBE master key is held by the trusted coordinator, and it is always able to decrypt any blinding factor in the public logs with a valid user identity. The adoption of IBE also reduces the overhead of key management for the IdP.

Improved Design. When encrypting the blinding factor s, I derives A's IBE public key and uses it in the encryption of $\mathbb{E}^{\mathcal{I}}_A(s)$. So A is able to decrypt s by his own IBE private key, which is inherently escrowed to the coordinator. However, for the user identity is not in the ticket entry, the coordinator has to decrypt $\mathbb{E}^{\mathcal{I}}_A(s)$ by exhaustively enumerating the user identities, especially for an IBE scheme with ACI-KGC [24], [47], where the ciphertext keeps indistinguishable even for the PKG who owns the IBE master key. We improve it to accelerate the dispute resolution, and $\mathbb{E}^{\mathcal{I}}_A(s)$ is generated along with another deterministically-encrypted copy $\mathbb{E}^{\mathcal{D}}_C(s)$.

Transparency against the Intelligent IdP. If a user suspects there exist fraudulent tickets labeled with his account, he follows the process described above to detect fraudulent tickets. After trying to decrypt the secret blinding factors of all suspected ticket entries with his IBE private key, if the user still has doubts, he initiates a *dispute resolution* process to the coordinator. Note that to initiate the process, the user needs to show some reasonable evidence to support his doubt, e.g., an abnormal log in the RP system.

Then, the coordinator examines the evidence to decide whether the dispute resolution process shall continue and coordinates the process if the evidence is reasonable. For a suspected ticket entry $\{\mathbb{S}_L(\mathbb{B}(m, s)), Sig_{I,m}, \mathbb{E}^{\mathcal{I}}_A(s), \mathbb{E}^{\mathcal{D}}_C(s)\}$ within the period of dispute, the coordinator recovers s from $\mathbb{E}^{\mathcal{D}}_C(s)$, and then uses s to recover m and $Sig_{L,m}$. Meanwhile, it will also generate the user's IBE private key to decrypt $\mathbb{E}^{\mathcal{I}}_A(s)$, to verify the correctness of this ticket entry (i.e., a valid user identity in m and the identical s in $\mathbb{E}^{\mathcal{D}}_C(s)$ and $\mathbb{E}^{\mathcal{I}}_A(s)$). Any failure implies a malicious IdP.

The coordinator obtains a valid ticket, if and only if the decrypted s is the correct blinding factor. Then, if this ticket is labeled with the disputer A's account, the coordinator sends it to A to allow him to continue the fraudulent-ticket detection as described above; otherwise, if all inquired tickets are associated with other users but not A, the coordinator will send nothing to A and the dispute resolution terminates.

4. Ticket Entry with Alias. The ticket entries in the public log contain blindly-signed signatures and encrypted (or blinded) tickets, which provide no plain information about the user for whom the ticket is issued. Therefore, a user who suspects fraudulent tickets are issued under his account has to try all tickets that are created in the suspected period, which is very inefficient. We adopt a Bloom filter $\mathbb{F}(\cdot)$ to generate aliases for users and record the aliases along with tickets issued under the enclosed accounts in the public logs. As a result, a user and the coordinator only need to search certain entries with his alias, in the fraudulent-ticket detection and the dispute resolution.

Improved Design. For each user A, the IdP calculates his alias $N_A = \mathbb{F}(A)$. When the user requests an SSO ticket, I sends this alias $\mathbb{F}(A)$ along with the encrypted or blinded data $\{\mathbb{B}(m, s), Sig_{I,m}, \mathbb{E}^{\mathcal{I}}_A(s), \mathbb{E}^{\mathcal{D}}_C(s)\}$ to L. Accordingly, the ticket entry consists of $\{\mathbb{F}(A), \mathbb{S}_L(\mathbb{B}(m, s)), Sig_{I,m}, \mathbb{E}^{\mathcal{I}}_A(s), \mathbb{E}^{\mathcal{D}}_C(s)\}$.

Efficiency and Privacy. The alias works as an index to facilitate the ticket search. In particular, a user only needs to check the ticket entries with his alias, instead of all the tickets in the suspicious period. The aliases in ticket entries leak some user privacy, for it provides a way to associate the sign-on activities of a certain user. If an attacker knows the user's account (or PPIDs in the following improved scheme), he could derive the user's sign-on pattern to some extent – the aliases are generated by the Bloom filter, so the user is identified but with a false positive rate. The privacy leakage is (*a*) controlled by the false positive rate of the adopted Bloom filter, and (*b*) limited to the approximate occurrence time of sign-on activities, but without the target RP that the user signs onto.

5. Blindly-Signed Transparent PPID-Ticket. The user accounts in tickets enable curious RPs to cooperatively track a user's network activities, so PPIDs are enclosed instead of accounts in existing SSO services. However, it is difficult and inconvenient for a user to remember all the randomly-generated PPIDs, which are needed to search for tickets in the logs. In our scheme, the PPID is deterministically computed

based on the user account and the target RP.

Since we do not want a user to hold more secrets (e.g., an extra symmetric key to encrypt A and R_i to compute the PPID), deterministic public-key encryption is adopted again to generate PPIDs. The corresponding private key is also held by the trusted coordinator. A user and the IdP compute the identical PPIDs, while the coordinator is able to decrypt the PPIDs in the dispute resolution.

Because a deterministic encryption scheme cannot protect the plaintext when the plaintext space is small [26], an extra random secret shared only between the user and the IdP is needed as the input of deterministic public-key encryption, and then the plaintext space becomes large enough. The hash value of A's IBE private key is used for this purpose – this design does not put extra burdens on the user or break the confidentiality of private keys. Although this secret can be alternatively used as a symmetric key to encrypt A and R_i to generate the PPID, in this way the coordinator cannot directly decrypt the data but has to try each user's secret one by one.

Improved Design. The PPID of A to visit R_i, denoted as PID_{A,R_i}, is computed as $PID_{A,R_i} = \mathbb{E}_C^{\mathcal{D}}(H(K_A^{\mathcal{I}}), A, R_i)$, where $H(\cdot)$ is a one-way hash function and $K_A^{\mathcal{I}}$ is A's IBE private key. So the SSO ticket and the alias are revised accordingly as $m = \{PID_{A,R_i}, R_i, o\}$ and $N_{A,R_i} = \mathbb{F}(PID_{A,R_i})$.

Deterministic and Recoverable PPID. The deterministic generation of PPIDs eliminates the user's burden to remember every PPID, especially in the fraudulent-ticket detection. Because only the PPID is enclosed in SSO tickets, the user identity needs to be recovered from the PPID by the coordinator in the dispute resolution (i.e., by decrypting PID_{A,R_i}); so we do not recommend a one-way function to generate PPIDs. With the user identity recovered from a PPID, the coordinator will still be able to generate the correct IBE private key to verify the ticket entry, because only PPIDs (but not the user identity) are enclosed in tickets after PPIDs are supported.

Privacy and Transparency. The PPIDs enclosed in the tickets to visit RPs are different, so the curious RPs cannot collude to track the user's activities [2], [27]. Because a user is able to re-compute his PPID to visit any R_i, he can retrieve all the ticket entries with $\mathbb{F}(PID_{A,R_i})$ and try to decrypt $\mathbb{E}_A^{\mathcal{I}}(s)$ of these ticket entries to detect fraudulent tickets.

PPID Verification. When a user suspects there are any fraudulent tickets, he first signs onto the target RPs one by one, to verify each PID_{A,R_i} and check that he is signing on behalf of himself. If any PID_{A,R_i} does not match $\mathbb{E}_C^{\mathcal{D}}(H(K_A^{\mathcal{I}}), A, R_i)$ or he signs onto some RP on behalf of another user, the IdP is asserted to be compromised. Then the user will retrieve the ticket entries in the public logs based on the verified PID_{A,R_i}.

6. Ticket Entry Verified by RPs. Two forms of each ticket are processed: (*a*) *transparent ticket* presented to the RP, i.e., $\{m, Sig_{I,m}, Sig_{L,m}\}$; and (*b*) *blinded/encrypted transparent ticket* logged in the logs along with the user alias, i.e., $\{\mathbb{F}(PID_{A,R_i}), \mathbb{S}_L(\mathbb{B}(m,s)), Sig_{I,m}, \mathbb{E}_A^{\mathcal{I}}(s), \mathbb{E}_C^{\mathcal{D}}(s)\}$. As mentioned above, the log server might not record some tickets in the logs intentionally or unintentionally. In this scheme, the honest RPs act as auditors [19] to audit the operations of the log server. Two forms of each ticket are sent together to the RP: the first one is to allow the user to sign on and the blinded/encrypted transparent tickets are necessary to enable the audit by RPs against the logs.

Improved Design. The IdP only sends the blinded/encrypted form $\{\mathbb{F}(PID_{A,R_i}), \mathbb{S}_L(\mathbb{B}(m,s)), Sig_{I,m}, \mathbb{E}_A^{\mathcal{I}}(s), \mathbb{E}_C^{\mathcal{D}}(s)\}$ to R_i, along with the blinding factor s. Then, R_i un-blinds $\mathbb{S}_L(\mathbb{B}(m,s))$ by itself to obtain the transparent ticket. Note that these messages are transmitted over secure channels (e.g., HTTPS) as in the original SSO protocols, and s is disclosed to A and R_i only.

Ticket Verification. The RP un-blinds the ticket by itself, and verifies two signatures on the transparent ticket to allow the ticket holder to sign on as PID_{A,R_i}. The RP also verifies (*a*) $\mathbb{F}(PID_{A,R_i})$ is correctly computed, and (*b*) the blinding factor s is escrowed to the coordinator by encrypting s into $\mathbb{E}_C^{\mathcal{D}}(s)$ again and comparing the results. $\mathbb{E}_C^{\mathcal{D}}(s)$ in the ticket entry does not leak any information on s, when s is randomly selected within a large space. Probabilistic encryption of s does not work, as the result of probabilistic encryption cannot be verified by the RPs during the online sign-on process, without the participation of the offline coordinator. If any of these verifications fail, the RP rejects this ticket immediately.

Audit against the Public Log. With the blinded/encrypted transparent tickets, the RP further checks whether every received valid ticket is recorded in the public logs as a ticket entry (including the alias, the ticket content, the signatures, and the encrypted copies of s). If there is no such an identical ticket entry, it indicates a malicious IdP or a lazy log server. Note that these audits may be conducted for a batch of tickets.

Similar as certificate transparency [19], the log server builds a Merkle hash tree over all ticket entries. By comparing the root node of the Merkle hash tree and requesting the audit path for each received valid ticket, the trusted RP checks whether the logs are append-only and the corresponding ticket entry exists in the logs or not. The Merkle audit path of a ticket entry, is the shortest list of additional nodes in the Merkle hash tree to compute the root node [19].

Privacy in the Dispute Resolution. Due to the verification by RPs, there is no fake alias in the ticket entries. So, in the dispute resolution, only the ticket entries including the same alias as the disputer, instead of all tickets in the suspected period, will be un-blinded (or decrypted) by the trusted coordinator.

D. Ticket Transparency in Full View

After the above step-by-step analysis, we present ticket transparency in full view.

Initialization Before the SSO service starts, the IdP generates its key pair to sign tickets, and the log server also generates its key pair to blindly sign tickets. The coordinator initializes its key pair of deterministic public-key encryption. All these public keys are publicly known. The coordinator also initializes the IBE parameters: the IBE public parameters are also publicly known, and the IBE master key is held by the coordinator. The Bloom filter is decided by the SSO service provider. The initial Merkle hash tree of logs is empty, and the root node stored on every RP is initialized as null.

When a user joins, he registers his account in the IdP, and applies for his IBE private key from the coordinator. Alternatively, a user may apply for the IBE private key, only when he suspects any fraudulent tickets labeled with his account and wants to search for the suspected tickets. Once A registers his account, the IdP requests $H(K_A^{\mathcal{I}})$ from the coordinator, to compute the user's PPIDs.

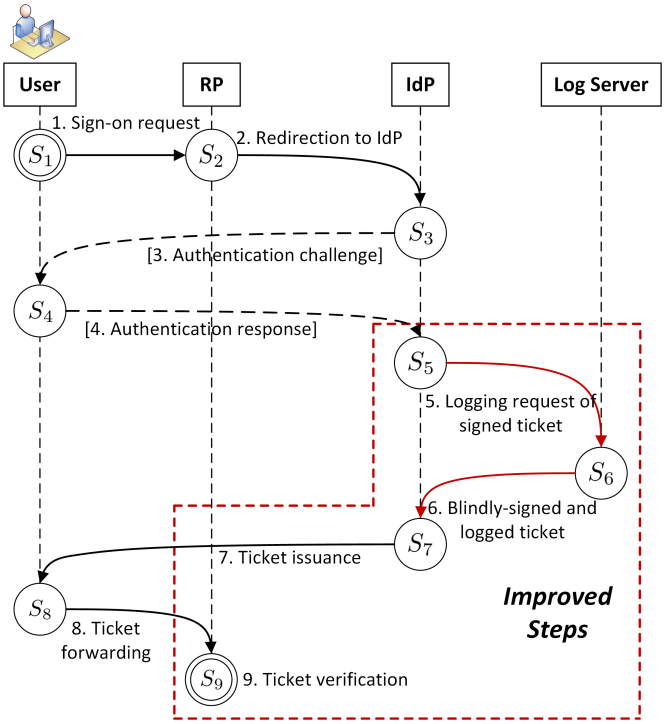

Fig. 1. Sign-On with Ticket Transparency

Sign-On When A attempts to sign onto R_i, he is redirected to I by R_i. After A is authenticated, I signs a ticket $\mathbb{S}_I(m) = \{m, Sig_{I,m}\}$, where $m = \{PID_{A,R_i}, R_i, o\}$ and $PID_{A,R_i} = \mathbb{E}_C^{\mathcal{D}}(H(K_A^{\mathcal{I}}), A, R_i)$. I blinds m into $m' = \mathbb{B}(m, s)$ with a blinding factor s, and computes $\mathbb{E}_A^{\mathcal{I}}(s)$ and $\mathbb{E}_C^{\mathcal{D}}(s)$. Then, I sends $\{N_{A,R_i}, \mathbb{B}(m, s), Sig_{I,m}, \mathbb{E}_A^{\mathcal{I}}(s), \mathbb{E}_C^{\mathcal{D}}(s)\}$ to L, where $N_{A,R_i} = \mathbb{F}(PID_{A,R_i})$.

L blindly signs m' into $\mathbb{S}_L(m') = \{m', Sig_{L,m'}\}$ and returns it to I. In the meantime, L creates a ticket entry of $\{N_{A,R_i}, \mathbb{S}_L(\mathbb{B}(m, s)), Sig_{I,m}, \mathbb{E}_A^{\mathcal{I}}(s), \mathbb{E}_C^{\mathcal{D}}(s)\}$ and records it chronologically in the public logs.

I sends $\{N_{A,R_i}, \mathbb{S}_L(\mathbb{B}(m, s)), Sig_{I,m}, \mathbb{E}_A^{\mathcal{I}}(s), \mathbb{E}_C^{\mathcal{D}}(s), s\}$ to A, who forwards it to R_i. R_i un-blinds $\mathbb{S}_L(\mathbb{B}(m, s))$ into $\mathbb{S}_L(m)$ and verifies two signatures on the transparent ticket (and conducts other verifications as specified in the original SSO protocols). R_i also verifies (a) N_{A,R_i} by computing $\mathbb{F}(PID_{A,R_i})$, and (b) $\mathbb{E}_C^{\mathcal{D}}(s)$ by encrypting s using the coordinator's pubic key. Then, it allows the ticket holder to sign on as PID_{A,R_i} after these verifications. These messages are kept for the later audits.

Figure 1 illustrates the sign-on process with ticket transparency, Steps 3 and 4 are optional if the user has been authenticated. Compared with the standard OpenID Connect sign-on process, the steps improved by ticket transparency, marked with red lines, include: (a) I calculates some elements of a ticket entry and sends them to L; (b) L blindly signs the ticket, returns it to I and records it in the public logs; (c) R verifies the extra signature, the alias and the encrypted blinding factors. The communications among I, A and R_i are over secure channels such as HTTPS, as in the original SSO protocol.

Audit R_i regularly requests the root node of the Merkle hash tree from L, each leaf node of which corresponds to a ticket entry of the public log. It checks whether the logs are append-only; i.e., the root node it stored corresponds to a sub-tree of the current Merkle hash tree. Then, the up-to-date root node is updated on R_i.

R_i requests the audit path from L, for each received valid ticket $\{N_{A,R_i}, \mathbb{S}_L(\mathbb{B}(m, s)), Sig_{I,m}, \mathbb{E}_A^{\mathcal{I}}(s), \mathbb{E}_C^{\mathcal{D}}(s)\}$. I or L is asserted to be faulty, if R_i does not receive a valid audit path from L.

Ticket Search and Dispute Resolution When suspecting there are any fraudulent tickets, A first signs onto the target RPs, to verify each PID_{A,R_i} and check that he is signing on behalf of himself. If PID_{A,R_i} does not match $\mathbb{E}_C^{\mathcal{D}}(H(K_A^{\mathcal{I}}), A, R_i)$ or he signs onto some RP on behalf of another user, the IdP is asserted to be compromised. Then, A retrieves all ticket entries where $N = \mathbb{F}(PID_{A,R_i})$, during the interested period. A tries these entries one by one, i.e., attempts to decrypt s and then un-blinds a ticket. If there is a ticket enclosing his PPIDs but without his participation, it is detected as fraudulent. A may fail to decrypt some entries, due to the false positive rate of $\mathbb{F}(\cdot)$.

If A still suspects fraudulent tickets after trying all entries with $\mathbb{F}(PID_{A,R_i})$ as above, he proposes a dispute with some reasonable evidence supporting this suspect. After the coordinator (a) verifies the user's identity and PPIDs (i.e., $PID_{A,R_i} = \mathbb{E}_C^{\mathcal{D}}(H(K_A^{\mathcal{I}}), A, R_i)$) and (b) examines this dispute and the evidence, for each suspected ticket entry where $N = \mathbb{F}(PID_{A,R_i})$ but A cannot un-blind the ticket, the coordinator decrypts $\mathbb{E}_C^{\mathcal{D}}(s)$, uses s to un-blind the message, and also decrypts $\mathbb{E}_A^{\mathcal{I}}(s)$ according to the user identity on the ticket: if the result is not a valid transparent ticket or the IBE decryption fails, it is asserted that I conducted something malicious. Otherwise, if PID_{A,R_i} is labeled in this ticket correctly, the coordinator sends the ticket to A, which indicates something abnormal with A's IBE private key; for example, it may result from a corrupted IBE private key. If it is a valid transparent ticket for another user, nothing is disclosed to A.

IV. Security Analysis and Discussion

This section analyzes ticket transparency in terms of correctness and privacy. We then compare it with privacy-preserving certificate transparency.

A. Correctness

The correctness is proved with the following sketch: (a) An SSO ticket is accepted by RPs, only if both the IdP and the log server have signed it; (b) Every SSO ticket accepted by RPs is recorded in the public logs; and (c) All elements of each ticket entry are verified by non-malicious entities (the user, the RPs,

or the coordinator), in the steps of sign-on, audit, ticket search or dispute resolution. Thus, the IdP and the log server have to follow their specifications; otherwise, any operation deviating from the specifications in the ticket creation or logging, will result in some verification failures eventually. So a user is able to detect fraudulent tickets by comparing the ticket entries in the logs and his history of sign-on activities, in the ticket search or the dispute resolution.

It is easy to verify that, with our scheme, an SSO ticket is accepted by RPs only if both the IdP and the log server have signed it, because the honest RPs verify two signatures on each ticket before accepting the ticket in the process of sign-on. Next, every ticket accepted by RPs is recorded in the logs as an entry and $N_{A,R_i} = \mathbb{F}(PID_{A,R_i})$; otherwise, if such a ticket is not recorded in the logs, the log server is detected as faulty by RPs in the audits. This is ensured by the verifications and audits by RPs. So a user is able to retrieve all his tickets among the ticket entries where $N_{A,R_i} = \mathbb{F}(PID_{A,R_i})$ and $PID_{A,R_i} = \mathbb{E}_C^{\mathcal{D}}(H(K_A^{\mathcal{I}}), A, R_i)$ in the ticket search.

All elements of each ticket entry are verified by the user, the RPs, or the coordinator. For each valid transparent ticket $\{m, Sig_{I,m}, Sig_{L,m}\}$, a corresponding ticket entry $\{N_{A,R_i}, \mathbb{S}_L(\mathbb{B}(m,s)), Sig_{I,m}, \mathbb{E}_A^{\mathcal{I}}(s), \mathbb{E}_C^{\mathcal{D}}(s)$ is verified by non-malicious entities as follows: (*a*) when A is signing on, the relationship of N_{A,R_i}, $\mathbb{S}_L(\mathbb{B}(m,s))$, $Sig_{I,m}$ and $\mathbb{E}_C^{\mathcal{D}}(s)$ is verified by the trusted RP, which un-blinds $\mathbb{S}_L(\mathbb{B}(m,s))$ to verify two signatures and N_{A,R_i}, and deterministically encrypts s by itself to verify $\mathbb{E}_C^{\mathcal{D}}(s)$; (*b*) the relationship of A, PID_{A,R_i}, N_{A,R_i}, $Sig_L(B(m,s))$, $Sig_{I,m}$, and $\mathbb{E}_A^{\mathcal{I}}(s)$ is verified by the user in the ticket search; and (*c*) the relationship of A, PID_{A,R_i}, N_{A,R_i}, $Sig_L(B(m,s))$, $\mathbb{E}_A^{\mathcal{I}}(s)$ and $\mathbb{E}_C^{\mathcal{D}}(s)$, is verified by the coordinator in the dispute resolution. Specially, a mismatching blinding factor will not happen for a ticket which has been verified and accepted by the RP, so that the user will always be able to decrypt all such ticket entries; sometimes, in the dispute resolution, the coordinator may find that the user identity does not match the ticket (i.e., A's IBE private key cannot decrypt the correct s) due to the false positive rate of Bloom filters. Besides, with the audit by RPs, nobody can tamper with a ticket entry after it has been verified.

In summary, the IdP cannot conceal the existence of a ticket if accepted by RPs. So a ticket issued without the user's participation or authorization, will be detected finally in the ticket search or in the dispute resolution.

B. Privacy

First of all, no *extra* user privacy is leaked to the IdP, compared with original SSO protocols. The IdP is always aware of the users' all sign-on activities, with or without ticket transparency. At the same time, PPIDs are supported against curious RPs, who track the user's activities by linking the identities in tickets across multiple RPs. It is computationally impossible for an RP to infer any information on the user identity A based on $PID_{A,R_i} = \mathbb{E}_C^{\mathcal{D}}(H(K_A^{\mathcal{I}}), A, R_i)$.

On the other hand, with ticket transparency, the privacy on a user's sign-on activities is leaked to some extent through public logs as follows. For each ticket entry $\{N_{A,R_i}, \mathbb{S}_L(\mathbb{B}(m,s)), Sig_{I,m}, \mathbb{E}_A^{\mathcal{I}}(s), \mathbb{E}_C^{\mathcal{D}}(s)\}$ in the logs, all elements except N_{A,R_i}, are meaningless (i.e., $Sig_{I,m}$ and $Sig_{L,m'}$) or encrypted/blinded (i.e., $\mathbb{B}(m,s)$, $\mathbb{E}_A^{\mathcal{I}}(s)$, and $\mathbb{E}_C^{\mathcal{D}}(s)$). These encrypted/blinded elements are decrypted/unblinded only by the user or the coordinator.

Next, we compare different scenarios when the Bloom filter is adopted (i.e., N_{A,R_i} appears in the ticket entry) or not. In the case of no dispute, if ticket transparency works *without* the Bloom-filtered aliases, there is not any extra privacy leaked, compared with the original SSO protocols, for only blinded/encrypted data are in the public logs. However, in the dispute resolution for A, the detailed sign-on activities of all users other than A during the interested period, have to be disclosed to the trusted coordinator.

After the Bloom filter is adopted to generate aliases, some privacy is leaked in the case of no dispute, while the privacy disclosed to the coordinator is relieved. A curious user can learn another user's history of sign-on activities but with a false positive rate (i.e., the false positive rate of $\mathbb{F}(\cdot)$), provided that the attacker knows the victim's PPIDs. Note that a PPID only appears in the communications among the user, the IdP and the target RP. Moreover, this privacy leakage is limited to the occurrence of sign-on activities but no information about the RP that is signed onto. Meanwhile, in the dispute resolution, only the sign-on activities of other users who have the same aliases as the disputer, are disclosed to the coordinator.

In summary, the quantity of the user privacy leaked in ticket transparency depends on the adopted Bloom filter. In general, a Bloom filter with a higher false positive rate, protects more user privacy in the case of no dispute but leaks more to the coordinator in the dispute resolution. In the extreme scenario, we may choose the Bloom filter with the maximum false positive rate (i.e., $\alpha = 1$ and it is a constant function), and actually no alias exists in the ticket entries. The Bloom filter shall be chosen carefully, because its false positive rate is a trade-off of efficiency and privacy.

C. Comparison with Privacy-Preserving Certificate Transparency

Privacy-preserving certificate transparency is proposed in two aspects [88]. The first design enables an auditor to prove that a log server does not record a certificate in the logs while it has received the corresponding SCT, in a zero-knowledge means (i.e., the SCT is not disclosed). It happens when the log server misbehaves. Although a ticket entry of ticket transparency consists of only the alias and other encrypted (or blinded) elements, this design can be integrated into our scheme, and an RP reports the log server's misbehavior without disclosing the unrecorded ticket entry.

The second privacy-preserving design [88] is to protect private (sub)domains in public logs – the public parent domain and a commitment [89] to the private subdomain appear in the SCT and also the logs. Then, in TLS negotiations the web server de-commits the commitment when presenting the certificate and the SCTs, so that browsers verify the private domain. The commitments in privacy-preserving certificate transparency take the same role as the blind signatures in ticket

transparency – the privacy information is hidden (or blinded) when sent to the log server and recorded in the public logs, and it is de-committed (or un-blinded) when presented to the verifier (i.e., browser or RP).

However, as described in Section III, blind signatures do not completely satisfy the requirements of ticket transparency well. This means that commitments cannot completely satisfy these requirements, either. So we have to utilize IBE to protect blinding factors to facilitate the ticket search, while [88] requires the domain owner to check the number of certificates issued for the parent domain. Bloom filters are adopted to generate aliases to filter out irrelevant ticket entries, while the owner of parent domains helps the private-subdomain owner to find all certificates [88]. But this strategy does not work for SSO tickets, because a user's account or PPID shall be protected as a whole and cannot work as split parts.

V. Implementation and Performance Evaluation

We implemented the prototype system and experimentally measured the performance.

A. Implementation

Original OpenID Connect We first build an SSO system, compatible with OpenID Connect. OpenID Connect provides SSO services with three flows [2]: implicit flow, authorization code flow, and hybrid flow (i.e., a mix-up of the previous two). In Section III, ticket transparency is described with the implicit flow, where the SSO ticket (i.e., id-token) is directly forwarded to the RP; however, in the authorization code flow which is the most commonly-used in practice, the RP firstly receives an authorization code forwarded by the user, and then uses this code to request an id-token from the IdP. Either in the implicit flow or in the authorization code flow, ticket transparency enhances the accountability of SSO ticket issuance (i.e., id-token signing) and the security analysis still holds.

In the prototype system, we run MITREid Connect as the IdP, and set up an RP based on Connect2id Nimbus. A user signs onto the RP through browsers following the authorization code flow of OpenID Connect.

OpenID Connect with Ticket Transparency We improve the SSO system with the following open-source tools to support ticket transparency: (*a*) Blind signature, deterministic public-key encryption and other cryptographic primitives are offered by Bouncy Castle; (*b*) IBE is implemented by Stanford IBE Library 0.72; and (*c*) the Bloom filter is borrowed from Guava. MITREid Connect and Connect2id Nimbus are modified accordingly, and we build the log server based on the open-source Merkle hash tree [90]. Besides, the processes of ticket search and dispute resolution are implemented by a coordinator and a client tool. All these functions are implemented by Java, except Stanford IBE Library is a C-language package.

The cryptographic schemes are as follows. 2048-bit RSA works as (*a*) the "plain" signature algorithm [91] by the IdP, (*b*) the blind scheme [23] by the log server, and (*c*) the deterministic encryption algorithm [26] to generate PPIDs and encrypt blinding factors. The Boneh-Franklin IBE scheme [24] with 1024-bit p and 224-bit q [92], [93] is adopted, for the same level of security strength. Besides, in the adoption of blind signatures, we do not directly blind the to-be-signed message because it is too long to be processed in one signing operation. Instead, we symmetrically encrypt the message using the blinding factor as an AES key, and the blinding factor blinds the SHA-256 hash value of the message before it is signed. The false positive rate of the Bloom Filter is 0.01. The Merkle hash tree adopts SHA-256. The communications among entities are protected by TLSv1.2.

The ticket verification by RPs in the sign-on process is further optimized as follows. The RP accepts a ticket, after un-blinding it and verifying two signatures on it (and other fields as specified in the original SSO protocols). These operations have prevented *external* attackers effectively. To speed up the sign-on process, the verifications of $\mathbb{F}(PID_{A,R_i})$ and $\mathbb{E}_C^{\mathcal{D}}(s)$ against faulty IdPs and log servers will be conducted by the RP later in the audit process against public logs.

These efforts are finished by about 3,000 lines of Java, including the modifications of MITREid Connect and Connect2id Nimbus, and the implementations of a log server and the coordinator, in addition to open-source toolkits.

B. Performance Evaluation

Three entities (i.e., the IdP, the log server and an RP) are connected in an isolated 1Gbps Ethernet LAN. The IdP works with Ubuntu 20.04 on Intel i7-4770S CPU (3.1GHz) and 8GB RAM, the log server runs Windows 10 on Intel i7-9750H CPU (2.6GHz), while the RP's configuration is Ubuntu 16.04 on Intel i7-3770 CPU (3.4GHz) and 8GB RAM. A user accesses this SSO prototype system through Firefox, with Ubuntu 16.04 on Intel i5-6500 CPU (3.2GHz). The offline coordinator is implemented with Ubuntu 16.04 on Intel i5-6500 (3.2GHz).

1) Sign-On: We compared the original OpenID Connect process and the process with ticket transparency. The time cost of the sign-on process is measured, starting when the RP sends the authorization-code request and ending when the RP finishes the ticket verification. We ran each process and the main cryptographic computations for 1,000 times. TLS session resumptions [94] happen between a pair of entities, so the cost of TLS handshakes is not included. The user has been authenticated before these measurements.

TABLE I
THE TIME COST OF SIGN-ON

Process / Operation	Time (ms)
Original OpenID Connect sign-on, in total	38.88
Ticket signing, by the IdP	8.60
IdP-signature verification, by the RP	0.29
OpenID Connect with ticket transparency sign-on, in total	87.81
Ticket signing, by the IdP	8.60
IBE encryption of blinding factor, by the IdP [*]	24.62
Message AES encryption, by the IdP [*]	0.05
Message blinding, by the IdP [*]	0.08
Ticket blind signing, by the log server [*]	9.05
Message AES decryption, by the RP [*]	0.05
Log-blind-signature un-blinding, by the RP [*]	0.33
Log-signature verification, by the RP [*]	0.25
IdP-signature verification, by the RP	0.29

The average time costs are shown in Table I. The original OpenID Connect requires 38.88ms to sign onto the target RP, while the system with ticket transparency takes 87.81ms. Some expensive pre-computations are not listed in Table I, such as RSA deterministic encryption to generate PPIDs, and Bloom filtering to generate aliases. The public keys of the IdP and the log server are pre-configured at the RP locally, so the RP does not request these public keys by online means.

The extra cost of 48.93ms introduced by ticket transparency mainly consists of (*a*) 34.43ms caused by extra computations, marked with '[*]' in Table I, and (*b*) one more round of network communication between the IdP and the log server.

The overheads introduced by ticket transparency in the sign-on process are acceptable. In practice, the total process of sign-on also includes the following *manual* operations, which are not measured in the above experiments. A user will be asked by the IdP to confirm the authorization to provide (privacy) information to target RPs, which usually consumes hundreds of milliseconds at least. The user is required to input his username/password or other credentials, when he is authenticated. So the overheads of ticket transparency do not significantly impact the user experience.

2) Audit, Ticket Search and Dispute Resolution: We measured the time costs of in the processes of audit, ticket search and dispute resolution, and Table II lists the average results.

TABLE II
THE TIME COSTS OF AUDIT, TICKET SEARCH AND DISPUTE RESOLUTION

Process / Operation	**Time (ms)**
Audit by the RP, in total	12.67
Verification of RSA-encrypted blinding factor	0.08
Verification of alias	0.11
Verification of audit path	0.02
Successful ticket search by the user, in total	20.67
Decryption of IBE-encrypted blinding factor	20.62
Decryption of AES-encrypted message	0.05
Failed ticket search by the user, in total	20.62
Decryption of IBE-encrypted blinding factor	20.62
Dispute resolution by the coordinator, in total	66.14
Decryption of RSA-encrypted blinding factor	12.40
Decryption of AES-encrypted message	0.05
Decryption of PPID	12.26
Generation of user IBE private key	20.81
Decryption of IBE-encrypted blinding factor	20.62

We prepared a Merkle hash tree with 5,000 ticket entries. The RP requests each ticket entry and also its audit path, one by one for these 5,000 entries. It averagely takes 12.67ms for the RP to request and verify the audit path, including the pre-verifications of deterministically-encrypted blinding factors and aliases. The main cost is the network communication with the log server: the communication needs 12.46ms, and the log server only needs 0.92ms to construct the audit path.

We measured the computations of each ticket search by the user. For a user may download all ticket entries in one operation, we do not measure the communication time. For each ticket entry, it takes 20.62ms on average for the user to decrypt the blinding factor, either successfully or unsuccessfully. If it is encrypted using the user's IBE public key, it then takes 0.05ms to symmetrically decrypt the message to obtain the plaintext ticket. Since all entries have been verified by the RPs, the user does not necessarily verify two signatures on the ticket.

It takes 66.14ms for the coordinator to process a ticket entry, in the dispute resolution. The coordinator decrypts the RSA-encrypted blinding factor, the plaintext ticket, and the PPID, re-generates the user's IBE private key, and decrypts the IBE-encrypted blinding factor. After all verifications, the coordinator will send the ticket to the disputer user or immediately terminate this process.

C. Optional Optimization

When encrypting the blinding factors, the IdP may derive a user's IBE public key based on his account and also a period of validity (e.g., the year) [24]. So the user updates his IBE private key periodically. Accordingly, the public logs of ticket entries are organized as multiple Merkle hash trees according to the occurrence time. Then, some trees of logs may be discarded and destroyed after a certain number of years, to mitigate the risk due to the unexpected exposure of IBE private keys.

When implementing the coordinator which is fully trusted in ticket transparency, (*a*) distributed PKGs [24], [85], accountable IBE [86], [87] and KGC-anonymous IBE [47] can be adopted to hold the IBE master key, and (*b*) threshold schemes [73], [74] can also be adopted to protect the private key of deterministic public-key encryption, to enhance the trustworthiness of the offline coordinator.

Customized Bloom filters may be adopted. Each user specifies the Bloom filter to generate his own alias, so that he is able to decide the false positive rate by himself based on his own privacy policy. Such a customized design requires the cooperation of the IdP and RPs, when a ticket entry is created and verified. The coordinator also needs to know the customized Bloom filter, in the dispute resolution.

VI. CONCLUSION

An SSO ticket issued by an IdP allows a user to sign onto numerous RPs on behalf of the account labeled in the ticket. Therefore, the online signing system of SSO services will become an attractive target of interests to adversaries so that they should not be fully trusted; Gold SAML attacks in the SolarWinds result in compromised IdP systems (specially, Microsoft Active Directory Federation Services). So we need a new security mechanism to detect and/or prevent fraudulent SSO tickets signed by the potentially compromised IdPs.

This paper proposes *ticket transparency*, *the first framework in the literature to provide the public accountability of IdPs*. With ticket transparency, each IdP-signed ticket is recorded in public logs, so that a user who suspects there are fraudulent tickets issued under his account is enabled to detect fraudulent tickets in the logs. To achieve transparency while mitigating the privacy leakage by the utilization of public logs, we integrate blind signatures, IBE, Bloom filters and deterministic public-key encryption in the design to balance transparency, privacy and efficiency in the SSO scenarios. We implemented the prototype system, and the performance analysis shows that ticket transparency introduces reasonable overheads.

REFERENCES

[1] D. Chu, J. Lin, F. Li, X. Zhang, Q. Wang, and G. Liu, "Ticket transparency: Accountable single sign-on with privacy-preserving public logs," in *15th International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2019, pp. 511–531.

[2] N. Sakimura *et al.*, "OpenID Connect Core 1.0," 2014, http://openid.net/specs/openid-connect-core-1_0.html.

[3] A. Nadalin *et al.*, "OASIS standard - Web services security: SOAP message security 1.1," 2006.

[4] M. Gudgin *et al.*, "W3C recommendation - SOAP version 1.2 part 1: Messaging framework (2nd edition)," 2007.

[5] D. Cooper *et al.*, "IETF RFC 5280: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile," 2008.

[6] Comodo Group Inc., "Comodo report of incident," 2011, https://www.comodo.com/Comodo-Fraud-Incident-2011-03-23.html.

[7] GlobalSign, "Security incident report," 2011, https://www.globalsign.com/resources/globalsign-security-incident-report.pdf.

[8] VASCO Data Security International Inc., "DigiNotar reports security incident," 2011, https://www.vasco.com/about-vasco/press/2011/news_diginotar_reports_security_incident.html.

[9] B. Morton, "More Google fraudulent certificates," 2014, https://www.entrust.com/google-fraudulent-certificates/.

[10] S. Reiner, "Golden SAML: Newly discovered attack technique forges authentication to cloud apps," 2017, https://www.cyberark.com/threat-research-blog/golden-saml-newly-discovered-attack-technique-forges-authentication-cloud-apps/.

[11] M. LaFerrera, "A Golden SAML journey: SolarWinds continued," 2021, https://www.splunk.com/en_us/blog/security/a-golden-saml-journey-solarwinds-continued.html.

[12] S. Reiner, "Golden SAML revisited: The Solorigate connection," 2020, https://www.cyberark.com/resources/threat-research-blog/golden-saml-revisited-the-solorigate-connection.

[13] NIST National Vulnerability Database, "CVE-2021-26715," 2021, https://nvd.nist.gov/vuln/detail/CVE-2021-26715.

[14] ——, "CVE-2021-27582," 2021, https://nvd.nist.gov/vuln/detail/CVE-2021-27582.

[15] M. Stepankin, "Hidden OAuth attack vectors," 2021, https://portswigger.net/research/hidden-oauth-attack-vectors.

[16] M. Ghasemisharif *et al.*, "O single sign-off, where art thou? An empirical analysis of single sign-on account hijacking and session management on the Web," in *27th USENIX Security Symposium*, 2018, pp. 1475–1492.

[17] C. Mainka *et al.*, "Do not trust me: Using malicious IdPs for analyzing and attacking single sign-on," in *1st IEEE European Symposium on Security and Privacy (Euro S&P)*, 2016, pp. 321–336.

[18] A. Yurchenko, "Golden SAML attack method used by APT group behind SolarWinds hack," 2021, https://socprime.com/blog/golden-saml-attack-method-used-by-apt-group-behind-solarwinds-hack/.

[19] B. Laurie, A. Langley, and E. Kasper, "IETF RFC 6962 - Certificate transparency," 2014.

[20] J. Amann *et al.*, "Mission accomplished? HTTPS security after DigiNotar," in *17th Internet Measurement Conference (IMC)*, 2017, pp. 325–340.

[21] J. Gustafsson *et al.*, "A first look at the CT landscape: Certificate transparency logs in practice," in *18th International Conference on Passive and Active Measurement (PAM)*, 2017, pp. 87–99.

[22] C. Nykvist *et al.*, "Server-side adoption of certificate transparency," in *19th International Conference on Passive and Active Measurement (PAM)*, 2018, pp. 186–199.

[23] D. Chaum, "Blind signatures for untraceable payments," in *Crypto*, 1982, pp. 199–203.

[24] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *Crypto*, 2001, pp. 213–229.

[25] B. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[26] M. Bellare *et al.*, "Deterministic and efficiently searchable encryption," in *Crypto*, 2007, pp. 535–552.

[27] P. Grassi *et al.*, "SP 800-63C - Digital identity guidelines - Federation and assertions," National Institute of Standards and Technology, Tech. Rep., 2017.

[28] R. Wang *et al.*, "Signing me onto your accounts through Facebook and Google: A traffic-guided security study of commercially deployed single-sign-on web services," in *33rd IEEE Symposium on Security and Privacy (S&P)*, 2012, pp. 365–379.

[29] J. Somorovsky *et al.*, "On breaking SAML: Be whoever you want to be," in *21st USENIX Security Symposium*, 2012, pp. 397–412.

[30] S.-T. Sun and K. Beznosov, "The devil is in the (implementation) details: An empirical analysis of OAuth SSO systems," in *19th ACM Conference on Computer and Communications Security (CCS)*, 2012.

[31] H. Wang *et al.*, "The Achilles heel of OAuth: A multi-platform study of OAuth-based authentication," in *32nd Annual Computer Security Applications Conference (ACSAC)*, 2016, pp. 167–176.

[32] ——, "Vulnerability assessment of OAuth implementations in Android applications," in *31st Annual Computer Security Applications Conference (ACSAC)*, 2015, pp. 61–70.

[33] Y. Zhou and D. Evans, "SSOScan: Automated testing of web applications for single sign-on vulnerabilities," in *23rd USENIX Security Symposium*, 2014, pp. 495–510.

[34] R. Wang *et al.*, "Explicating SDKs: Uncovering assumptions underlying secure authentication and authorization," in *22nd USENIX Security Symposium*, 2013, pp. 399–314.

[35] W. Li and C. Mitchell, "Analysing the security of Google's implementation of OpenID Connect," in *13th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2016, pp. 357–376.

[36] J. Navas and M. Beltran, "Understanding and mitigating OpenID Connect threats," *Computers & Security*, vol. 84, pp. 1–16, 2019.

[37] C. Baum *et al.*, "Pesto: Proactively secure distributed single sign-on, or how to trust a hacked server," in *5th IEEE European Symposium on Security and Privacy (Euro S&P)*, 2020, pp. 587–606.

[38] S. Liu *et al.*, "SGX-Cube: An SGX-enhanced single sign-on system against server-side credential leakage," in *16th International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2020, pp. 275–290.

[39] D. Fett *et al.*, "Analyzing the BrowserID SSO system with primary identity providers using an expressive model of the Web," in *20th European Symposium on Research in Computer Security (ESORICS)*, 2015, pp. 43–65.

[40] ——, "SPRESSO: A secure, privacy-respecting single sign-on system for the Web," in *22nd ACM Conference on Computer and Communications Security (CCS)*, 2015, pp. 1358–1369.

[41] K. Elmufti *et al.*, "Anonymous authentication for mobile single sign-on to protect user privacy," *International Journal of Mobile Communications*, vol. 6, no. 6, pp. 760–769, 2008.

[42] J. Han *et al.*, "A generic construction of dynamic single sign-on with strong security," in *6th International Conference on Security and Privacy in Communication Networks (SecureComm)*, 2010, pp. 181–198.

[43] J. Wang *et al.*, "Anonymous single sign-on schemes transformed from group signatures," in *5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, 2013, pp. 560–567.

[44] T.-F. Lee, "Provably secure anonymous single-sign-on authentication mechanisms using extended Chebyshev Chaotic Maps for distributed computer networks," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1499–1505, 2018.

[45] J. Han *et al.*, "Anonymous single-sign-on for n designated services with traceability," in *23rd European Symposium on Research in Computer Security (ESORICS)*, 2018, pp. 470–490.

[46] ——, "Anonymous single sign-on with proxy re-verification," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 223–236, 2020.

[47] S. Chow, "Removing escrow from identity-based encryption," in *12th International Conference on Practice and Theory in Public Key Cryptography (PKC)*, 2009, pp. 256–276.

[48] X. Bao *et al.*, "Towards the trust-enhancements of single sign-on services," in *3rd IEEE Conference on Dependable and Secure Computing (IDSC)*, 2019.

[49] SSL Shopper, "SSL certificate for mozilla.com issued without validation," 2008, https://www.sslshopper.com/article-ssl-certificate-for-mozilla.com-issued-without-validation.html.

[50] K. Wilson, "Distrusting new CNNIC certificates," 2015, https://blog.mozilla.org/security/2015/04/02/distrusting-new-cnnic-certificates/.

[51] StartCom Limited, "Critical event report," 2008, https://blog.startcom.org/wp-content/uploads/2009/01/ciritical-event-report-12-20-2008.pdf.

[52] B. Morton, "Public announcements concerning the security advisory," 2013, https://www.entrust.com/turktrust-unauthorized-ca-certificates.

[53] M. Zusman, "Criminal charges are not pursued: Hacking PKI," 2009, https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-zusman-hacking_pki.pdf.

[54] A. Langley, "Further improving digital certificate security," 2013, https://security.googleblog.com/2013/12/further-improving-digital-certificate.html.

[55] P. Eckersley, "A Syrian man-in-the-middle attack against Facebook," 2011, https://www.eff.org/deeplinks/2011/05/syrian-man-middle-against-facebook.

[56] C. Soghoian and S. Stamm, "Certified lies: Detecting and defeating government interception attacks against SSL," in *15th Financial Cryptography and Data Security Conference (FC)*, 2012, pp. 250–259.

[57] B. Dowling *et al.*, "Secure logging schemes and certificate transparency," in *21st European Symposium on Research in Computer Security (ESORICS)*, 2016, pp. 140–158.

[58] O. Gasser *et al.*, "In log we trust: Revealing poor security practices with certificate transparency logs and internet measurements," in *19th International Conference on Passive and Active Measurement (PAM)*, 2018, pp. 173–185.

[59] B. Li *et al.*, "Certificate transparency in the wild: Exploring the reliability of monitors," in *26th ACM Conference on Computer and Communications Security (CCS)*, 2019, pp. 2505–2520.

[60] ——, "The weakest link of certificate transparency: Exploring the TLS/HTTPS configurations of third-party monitors," in *18th IEEE International Conference on Trust, Security and Privacy in Computing And Communications (TrustCom)*, 2019, pp. 216–223.

[61] B. Laurie and E. Kasper, "Revocation transparency," 2012, http://sump2.links.org/files/RevocationTransparency.pdf.

[62] M. Ryan, "Enhanced certificate transparency and end-to-end encrypted mail," in *21st ISOC Network and Distributed System Security Symposium (NDSS)*, 2014.

[63] A. Singh *et al.*, "Certificate transparency with enhancements and short proofs," in *22nd Australasian Conference on Information Security and Privacy (ACISP)*, 2017, pp. 381–389.

[64] Z. Wang *et al.*, "Blockchain-based certificate transparency and revocation transparency," *IEEE Transactions on Dependable and Secure Computing*, accepted.

[65] Mozilla, "Binary transparency," 2017, https://wiki.mozilla.org/Security/Binary_Transparency.

[66] The Linux Foundation, "Sigstore: A non-profit, public good software signing & transparency service," 2021, https://sigstore.dev/.

[67] J. Beekman *et al.*, "Attestation transparency: Building secure Internet services for legacy clients," in *11th ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2016, pp. 687–698.

[68] M. Chase and S. Meiklejohn, "Transparency overlays and applications," in *13th ACM Conference on Computer and Communications Security (CCS)*, 2016, pp. 168–179.

[69] M. Melara *et al.*, "CONIKS: Bringing key transparency to end users," in *24th USENIX Security Symposium*, 2015, pp. 383–398.

[70] T. Pulls *et al.*, "Distributed privacy-preserving transparency logging," in *12th ACM Workshop on Privacy in the Electronic Society (WPES)*, 2013, pp. 83–94.

[71] R. Peeters and T. Pulls, "Insynd: Improved privacy-preserving transparency logging," in *21st European Symposium on Research in Computer Security (ESORICS)*, 2016, pp. 121–139.

[72] S. Matsumoto *et al.*, "CASTLE: CA signing in a touch-less environment," in *32nd Annual Computer Security Applications Conference (ACSAC)*, 2016, pp. 546–557.

[73] Y. Desmedt, "Society and group oriented cryptography: A new concept," in *Crypto*, 1987, pp. 120–127.

[74] V. Shoup, "Practical threshold signatures," in *EuroCrypt*, 2000, pp. 207–220.

[75] J. Jing *et al.*, "ARECA: A highly attack resilient certification authority," in *1st ACM Workshop on Survivable and Self-Regenerative Systems (SSRS)*, 2003, pp. 53–63.

[76] P. Erman *et al.*, "Preventing cryptographic key leakage in cloud virtual machines," in *23rd USENIX Security Symposium*, 2014.

[77] C. Wang *et al.*, "Privacy-preserving public auditing for secure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[78] ——, "Privacy-preserving public auditing for data storage security in cloud computing," in *IEEE INFOCOM*, 2010, pp. 525–533.

[79] Q. Wang *et al.*, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[80] Q. Liu *et al.*, "Consistency as a service: Auditing cloud consistency," *IEEE Transactions on Network and Service Management*, vol. 11, no. 1, pp. 25–35, 2014.

[81] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *14th ACM Conference on Computer and Communication Security (CCS)*, 2007, pp. 598–610.

[82] K. Bowers *et al.*, "Proofs of retrievability: Theory and implementation," in *ACM Workshop on Cloud Computing Security (CCSW)*, 2009, pp. 43–54.

[83] R. Houlihan *et al.*, "Auditing cloud service level agreement on VM CPU speed," in *IEEE International Conference on Communications (ICC)*, 2014, pp. 799–803.

[84] J. Kroll, "Outlining traceability: A principle for operationalizing accountability in computing systems," in *4th ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2021, pp. 758–771.

[85] A. Kate and I. Goldberg, "Distributed private-key generators for identity-based cryptography," in *7th International Conference on Security and Cryptography for Networks (SCN)*, 2010, pp. 436–453.

[86] V. Goyal, "Reducing trust in the PKG in identity-based cryptosystems," in *Crypto*, 2007, pp. 430–447.

[87] V. Goyal *et al.*, "Black-box accountable authority identity-based encryption," in *15th ACM Conference on Computer and Communications Security (CCS)*, 2008, pp. 427–436.

[88] S. Eskandarian *et al.*, "Certificate transparency with privacy," in *17th International Symposium on Privacy Enhancing Technologies (PETS)*, 2017, pp. 329–344.

[89] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Crypto*, 1991, pp. 129–140.

[90] S. Stefani, "merkle-tree-java," https://github.com/SimoneStefani/merkle-tree-java.

[91] RSA Laboratories, "PKCS #1 v2.2: RSA cryptography standard," EMC Corporation, Tech. Rep., 2012.

[92] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "SP 800-57 - Recommendation for key management - Part 1: General," National Institute of Standards and Technology, Tech. Rep., 2006.

[93] X. Boyen and L. Martin, "IETF RFC 5091: Identity-based cryptography standard (IBCS) #1: Supersingular curve implementations of the BF and BB1 cryptosystems," 2007.

[94] A. Arampatzis, "TLS Session Resumption," 2019, https://www.venafi.com/blog/tls-session-resumption.