

On Timed Zero Knowledge Proof (ZKP) Protocols (Preliminary Results)

T. C. Lam and Jyh-Charn Liu

Department of Computer Science, Texas A&M University

College Station, Texas 77843-3112

Email: brianlam@ieee.org and liu@cs.tamu.edu

Abstract—A major application of zero knowledge proof (ZKP) protocols is for authentication of anonymous users. Many ZKP protocols also control anonymity quota so that any excessive use of authorized tokens issued to the user (prover) will allow the authenticator (verifier) to decipher the user identity unambiguously. ZKP has great potentials to be used for creating spoof-free trust chains among hostile peer computing systems. Given the critical nature of trust chains, the temporal behaviors and the operational overheads of ZKP need to be adequately addressed. In this work-in-progress article, we report preliminary results on how to guarantee the security properties of timed ZKP (TZKP) protocols locked into the logical clocks for session based resource access. Preliminary findings suggest that security vulnerability can be prevented with reduced overheads after proper refinements to the disposable authentication protocols.

I. INTRODUCTION

With massive deployments of Internet applications, various Internet management functions, such as routing, DNS, and other name/number services need to be closely guarded to avoid application frauds and malicious acts that exploit the protocol weakness. When spoofing occurs, it implies that the existing identity services could have been compromised, and the contingent authentication procedures will need to be instantiated to restore the *trust chain*, so that peer nodes can guarantee that they possess provable identities issued by the recognized authority(ies).

A trust chain can typically be built by using traditional digital signatures, where a statement for delegation of trust is digitally signed in each authentication instance, and a series of signatures represent the hierarchy of trust relationships among different nodes. However, to verify a trust chain, an authenticator has to have prior knowledge on the public keys of all signers in the trust chain. The heavy reliance on the individual public key information makes it less flexible to be deployed in large scale for decentralized systems, where the *central authority* (CA) may stay *off-line* most of the time.

Zero knowledge proof (ZKP) protocols provide another way for authentication, which do not require the individual's public key or other data that can be used to distinguish an individual node from another. All nodes in a trust chain can remain anonymous, while having their identities verifiable. Given the minimum amount of identity information into the trust chain, it helps to prevent problems such as the passive analysis by adversaries or identity bias in the fairness protocols. However, most ZKP protocols are designed for human-user based appli-

cations, such as *e-cash*, *e-voting*, and little is considered about the operational overheads and the temporal behaviors for trust chain formation in distributed computing systems.

In the standard *three-move* ZKP protocol, a principal first *withdraws* from the CA a *token*, which contains the (secret) identity information of the principal. Then it acts as a prover (*P*) to ask for interaction with an *authenticator*, or commonly called a *verifier* (*V*), by sending it a *witness* message created from the token. *V* returns a randomly generated *challenge* message to *P*, and *P* must use the token and the challenge as part of its computing inputs to produce a *response* message, and return it to *V*. *V* knows that *P* has a valid token issued from the CA if it can use the messages (witness, challenge, response), collectively called the *credential*, and some CA's published data as inputs to produce a "YES" result from the publicly known *verifier function*. *P* can prove the legitimacy of its token to any parties without giving out its true identity.

A classic technique for quota control in ZKP protocols is by constructing a threshold-based function that can calculate the true identity from multiple copies of credentials when the quota is exceeded. Quota control has significant implications to the designs of *mutual exclusion* for critical resource, such as processor cycles, memory spaces, or conflict roles in role-based systems. Excessive access of critical resources via duplicates of authorization will lead to unexpected system states. Many distributed algorithms consider honest (deterministic) behaviors or fault tolerance (stable outcomes after detection and recovery), but identification of faulty sources, such as quota violators, is rarely addressed. Quota control offers additional protection to secure critical resources via accountable actions.

In spite of the well established cryptographic properties for authenticity, integrity, non-repudiation, anonymity quota control, *etc.*, most ZKP protocols focus on the spatial domain of trust chain formation, *e.g.*, delegation tree/network is built via propagation and proliferation of trust. Lacking notion of time in existing ZKP protocols make them suboptimal to be used for system management applications: once a token is issued, it remains valid indefinitely until its usage exceeds its quota, and when the quota is exhausted, a new token has to be withdrawn to avoid exposing secret identity information on further use. How to incorporate the notion of time into ZKP protocols plays a pivoting role on the understanding of their temporal behaviors. Such an understanding is crucial to

answer such questions as how to reset quota to initial system states without inducing withdrawals of new tokens; how to erase expired audit trail for detection of excessive access; how to identify a prover who intentionally gives out its token to unauthorized parties so that they can engage in multiple authentications; and how to support a prover to engage with multiple verifiers without sacrificing anonymity. Complexity of the problems is escalated by the fact that a principal may act as either P or V in concurrent sessions.

How to guarantee deterministic protocol outcome under varying network delays needs to be addressed step by step, with unambiguously defined system conditions. In this work, we report preliminary findings on the relationships between ZKP protocol behaviors and synchronization of multiple clocks, as a first step in the broad investigations of ZKP based system management protocols.

II. TIMED ZKP (TZKP)

Definition 1: Timed ZKP (TZKP) is a ZKP protocol whose tokens, credentials and other cryptographic primitives carry time information so that the correctness of operations, such as withdrawal, authentication, and anonymity quota control can be defined with respect to progression of *logical clocks*.

A logical clock is the reading of a local physical clock, which has a nearly constant frequency. The value and pace of a logical clock can be adjusted by system software based on the logical clocks of other nodes. A simplest example of TZKP is to add the notion of time-out to the ZKP data, *i.e.*, a token becomes void and needs to be reissued to a principal if it is not used for a predefined life cycle. This way, the quota control subsystem does not need to store ALL token or credential information indefinitely. On the other hand, for certain applications, this is highly desirable for a prover not to withdraw a new token to perform periodic authentication with the verifiers, in order to minimize withdrawal overhead.

Definition 2: In the *session based* TZKP system, the logical clock value in each prover is synchronized to the clock value in its verifier and optionally with the values of other provers under some known physical bounds. TZKP messages associated with each logical clock tick are guaranteed to be ordered within the boundaries of the logical clocks.

We assume that V serves as a *service provider* and P a *client* of the service. A logical clock tick between the client and the server is one TZKP session in which the client wants to be authenticated by the server to gain access to the service such as online game, database inquiry, *etc.* Among various alternatives, we focus on a *P-initiated* clock synchronization scheme, in which a new *authentication clock tick (session)* starts when P sends out the first TZKP message to V . Similar to the standard ZKP protocols, V gives P a challenge, and P must return a response for authentication. Once V produces a "YES" answer, it can give P access to the services. A subtle difference between TZKP and ZKP is that V has a published, periodically update unique *session ID* that must be used to produce P 's response. The session ID serves two purposes: global synchronization of concurrent sessions from multiple

P 's, and prevention of collusion among multiple P 's, when an authorized token is shared among them for excessive access of services. P can use the same token to access multiple V 's for different services. To guarantee the anonymity of a rule-abiding P , no session ID's in its access history to various V 's can be duplicated, to prevent collusion among V 's. After P is authenticated for a session, follow up actions may be taken to exchange other shared secrets for the particular session.

For space limitation, we focus on two major issues in this paper. The first issue is clock synchrony of TZKP message passing, *i.e.*, boundaries between logical clocks/sessions, to avoid unforeseeable vulnerabilities in the protocol designs. The second issue is how to minimize the computing and communication overheads of TZKP. A user needs to go through a similar authentication process using the same token to enter a new session. Typical ZKP protocol offers constructs that allow P to use a token within a finite quota without compromising its identity. This is very costly when resource accesses need to be done in sessions, because P will need to repeatedly withdraw tokens. We propose a novel solution to eliminate such overheads. By incorporating the session ID to the baseline ZKP constructs, P can produce valid credentials, but V cannot decipher P 's identity from the credentials produced in multiple sessions, each of which has a unique session ID. This way, P can access each session at most once, and V can decipher P 's identity only when it has multiple copies of credentials (exceeding quota) generated by a token in the same session.

Asynchrony of logical clock may cause inconsistent local views of session length, starting times, *etc.*, at boundaries of adjacent sessions, called *gap* and *crosstalk*.

Definition 3: Gap is a time interval of P (or V) that belongs to no session of P (or V), or a time interval in session i of V within which no message sent from session i of P can be received by V .

Definition 4: Crosstalk is a time interval of P (or V) within which a message sent in session i of P can be received in a gap or session j of V , where $i > j$ or $j > i$.

Suppose the session length from the local views of P and V in each of the n consecutive sessions is fixed to be $T_{S(P)}$ and $T_{S(V)}$, respectively, for convenience of discussions, and we let the first session begins at time t_0 , and the last session ends at time t_n . The expected outcome by using TZKP is a ration-like system which allows at most n legitimate accesses in n sessions of time (one access per session.) The existence of gaps, however, reduces the frequency of legitimate access because fewer than n accesses are allowed within the time interval $[t_0, t_0 + n \cdot T_{S(P)}]$ of P (or $[t_0, t_0 + n \cdot T_{S(V)}]$ of V .) Similarly, crosstalks penalize the effective time for legitimate accesses because it allows n accesses in less than $n \cdot T_{S(P)}$ (or $n \cdot T_{S(V)}$) of time. In addition to the above concerns about access patterns, crosstalks may also cause potential hazard to mis-ordering of messages across session boundaries. Ordering of message is important to many distributed algorithms. An early exposure of secret knowledge may increase the (Shannon) information for adversaries to launch attacks, while a belated delivery of message may lead to tardy delay of

authentication.

Gap and crosstalk in TZKP are caused by message delay, response time, virtual clock frequency and the request time for new session. We first consider the case when message delays between P and V are constant and the virtual clocks of P and V are fully synchronized, and then we will relax these conditions step-by-step. In session 0 (or the transient state when no session has been defined), P sends to V a new session request with a delay $T_{D_0(P,V)}$. Then, V takes a response time $T_{R_0(V)}$ to process the request message, and sends to P a message with a delay $T_{D_0(V,P)}$ to grant the session. P takes a response time $T_{R_0(P)}$ to process the grant message before it can start session 1. The total response time from P 's request being sent to session 1 being started is $T_{R_0(P,V)} = T_{D_0(P,V)} + T_{R_0(V)} + T_{D_0(V,P)} + T_{R_0(P)}$.

Our attempt is to find an optimal time for P to request the next session, so that there is no gap or crosstalk between the current session and the next session. If the response time is constant, then the optimal time for P to submit the request for session 2 is at $T_{I_1(P,V)} = T_{S_1(P)} - T_{R_1(P,V)}$ after the start of session 1, where $T_{S_1(P)}$ is the length of session 1 granted by V in session 0. When variable response time is considered, however, P needs to estimate this optimal time based on previous values. If the estimated time is $T_{I_1^+(P,V)} > T_{I_1(P,V)}$, then a gap occurs between session 1 and session 2, with the gap length $T_{I_1^+(P,V)} - T_{I_1(P,V)}$. Similarly, if the estimated time is $T_{I_1^-(P,V)} < T_{I_1(P,V)}$, then a crosstalk occurs between session 1 and session 2, with their common interval $T_{I_1(P,V)} - T_{I_1^-(P,V)}$. For underestimated value of the optimal time, V can minimize the crosstalk interval by adding extra delay to its response. For overestimated value, however, V can do nothing to glue the gap. Instead, P should avoid gap from happening by posting early request of new session.

Next, we consider the case when the virtual clocks of P and V run at different frequencies. Suppose P 's clock runs faster (at a higher frequency) than V 's clock. In session 0, V expects to start session 1 after certain delay it sends P the message to grant the new session. However, since P 's clock runs too fast, it starts session 1 earlier than V expected. Thus, the message sent at the very beginning in session 1 of P may crosstalk with session 0 of V . In addition, the length of a session is defined by the number of ticks in P 's and V 's logical clocks. Since P 's clock runs too fast, P 's session 1 has a shorter physical time interval than V 's. P finishes session 1 earlier than V expected, and V does not grant new session to P until V finishes its session. Thus, a gap is formed between session 1 and 2 of P .

Finally, we consider the case when message delays are variable and bounded. Suppose $E(T_{D(P,V)})$, $Max(T_{D(P,V)})$ and $Min(T_{D(P,V)})$ respectively denote the expected, maximum and minimum message delay. As depicted in Figure 1, crosstalks may occur at both boundaries of a session (of P and V). P 's session 1 crosstalks with V 's session 0 at the first interval of $T_{C_{1,0}(P)} = Max(T_{D(P,V)}) - Min(T_{D(P,V)})$. P 's session 1 crosstalks with V 's session 2 at the last interval of $T_{C_{1,2}(P)} = Max(T_{D(P,V)}) - E(T_{D(P,V)})$. V 's session 1 crosstalks with P 's session 0 at the first interval

of $T_{C_{1,0}(V)} = Max(T_{D(P,V)}) - E(T_{D(P,V)})$. V 's session 1 crosstalks with P 's session 2 at the last interval of $T_{C_{1,2}(V)} = E(T_{D(P,V)}) - E(T_{D(P,V)})$. The total length of crosstalk interval is minimized when $E(T_{D(P,V)}) = Max(T_{D(P,V)})$, so that crosstalk at the end of a session does not occur. However, we note that maximum delay of messages does not occur very often. If we set the expected delay to be the maximum delay, then all messages which have a delay smaller than the maximum delay will crosstalk at the beginning of the session for sure.

To better deal with the crosstalk and the gap problems, we propose a simple method for delay estimation and clock synchronization between P and V , as depicted in Figure 2. The basic idea is to estimate the current delay and clock rate based on previous round trip messages. When P sends a message to V , it also includes the number of ticks it takes since the last message received from V . Similarly, when V sends a message to P , it also includes the number of ticks it takes since the last message received from P . After two sets of round trip messages, P (or V) knows two linear equations, with several unknowns: duration of one P 's tick, duration of one V 's tick, and a number of message delays. The ratio between P 's tick vs. V 's tick, between P 's tick vs. expected delay, and V 's tick vs. expected delay can be well estimated by solving by these linear equations, assuming that the delays between adjacent messages are more or less the same.

III. LINKABILITY

Linkability refers to the ability for an authenticator to decide whether or not two credentials are produced by the same user, although the user identity may still be unknown. TZKP can be implemented by unlinkable schemes like those proposed in [1]. Unlinkability gives strong protection on P 's identity but imposes significant computing burdens for V to reinforce quota control. For system management functions, linkability, rather than unlinkability, is a highly desirable attributes, so that system events can be traced for various purposes. To protect P 's identity in authentication sessions, a unique session ID is used as a part of inputs to compute the credential. If the same session ID is used to produce two credentials generated by one token, then the user identity can be deciphered from the credentials. Otherwise, if distinct session ID's are used, then the user identity remains undecipherable.

In unlinkable schemes, such as [1], all users who access the authenticator "simultaneously" have to use the same session ID. Otherwise in an unlinkable scheme, the authenticator cannot tell whether or not the credentials are created by the same token. In contrast, this constraint is not required in linkable schemes. Different users who access the authenticator concurrently can use different session ID's. The relaxed requirement removes the overhead on co-operation or negotiation of a common session ID between independent users who just access the authenticator at the same physical time interval.

In this paper, we propose a suite of linkable schemes revised from the *disposable authentication* protocol [2] for TZKP. Disposable authentication is a class of one-time ZKP

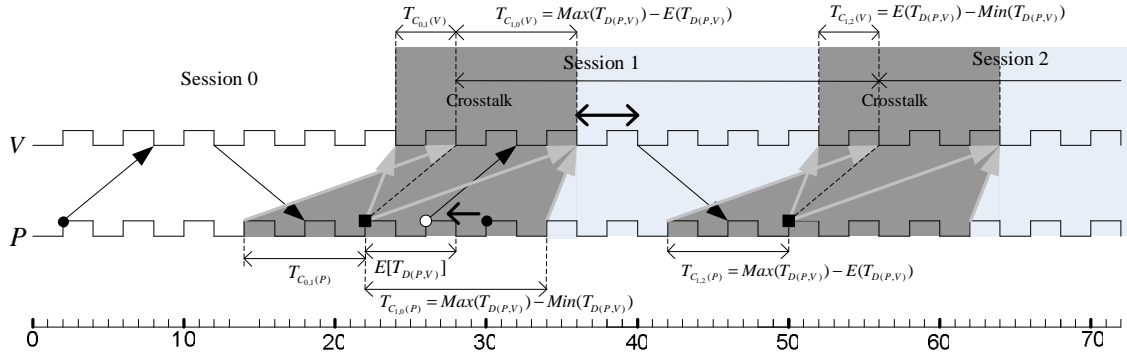


Fig. 1. Crosstalks in TZKP

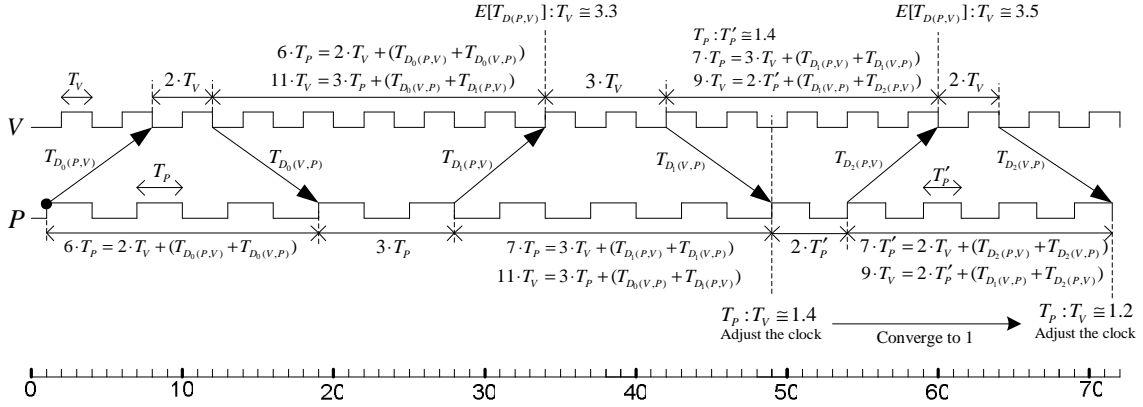


Fig. 2. Synchronization of clocks

originated for e-cash protocol. In its original construct, a secret key k is included in a token, which public counterpart K is committed to the bank (CA) at the withdrawal phase. k is used by P to encrypt the user identity to the credential, and K is used by V to verify that the user identity has been encrypted to the credential. The deciphering condition of user identity in this construct is as follows: the user identity can be deciphered from the credential if, and only if, the same k (and hence the same K) is used to create the credential. Our main idea is to separate k from the internal construct of the token, and give users the freedom to select k in the authentication phase. K is then committed to the authenticator, which plays the role as session ID in TZKP. Let f, F, F', D , and G be some polynomial time computable functions, and f is one-way. The three-move protocol is sketched below: First, P randomly generates k which has not yet been used before. P computes $K = F(x, k) = F'(m, k)$, where x is a secret value that can be mapped to P 's identity, and m is the public part of the token (m, x) , which satisfies $m = f(x)$. Then P sends V a witness message $W = (m, K)$. Upon receiving W , V checks whether or not m has been received before in the session. If yes, that means this is the same principal, V needs to make sure that the proposed K is also the one received before. Then V sends back P a randomly generated challenge E . P replies V by a response $Y = D(x, k, E)$. The protocol completes after V validates the

credential by checking that $G(W, E, Y) = \text{"YES"}$.

IV. CONCLUSION

TZKP represents a simple yet significant augmentation of ZKP for establishing trust chains in presence of repeated spoofing attacks. This paper reports the first, basic step to understand the relationship between logical clocks and message passing in ZKP protocols. Preliminary results suggest that TZKP can significantly reduce the overheads in token withdrawals without compromising the cryptographic properties.

V. ACKNOWLEDGEMENT

This work is supported in part by an NSF CNS-0530210 and DUE-0516825.

REFERENCES

- [1] I. Damgård, K. Dupont, and M. Ø. Pedersen, "Uncloneable group identification." *Cryptology eprint Archive*, report 2005/170 (2005).
- [2] T. Eng and T. Okamoto, "Single-term divisible electronic coins." In *Advances in Cryptology EUROCRYPT '94* (1994), 331-323.