Comments on Recent Advances in Cryptoanalysis of URSA

"In theory there is no difference between theory and practice. In practice there is..."

–Bruce Schneier in *Secrets and Lies* [8],2000

1 Introduction

Membership control (or access control) is a challenging problem in self-organized mobile ad-hoc networks (MANETs), notably because these networks typically do not involve a centralized entity that is trusted by all network participants. Moreover, even if such a central authority exists, it not only poses the security threat of a single point of failure, but also impedes the scalability and availability of the security services, because the mobile nodes have to frequently locate and communicate with this centralized entity over multi-hop, highly dynamic, and bandwidth-constrained wireless channel. As a result, the existing access control mechanisms that follow a client/server architecture cannot be directly applied in the context of MANETs.

In earlier publications [5, 3, 4], we have proposed URSA, a peer-to-peer membership control framework for MANETs that addresses the above challenges. Our contributions in these work are three-fold. First, we proposed the URSA architecture that exempts the necessity of centralized entities by delegating the membership control authority equally to all nodes in the network. Second, we presented a protocol suite that implements this architecture, and evaluated its network performance through both simulations and implementation. Third, we proposed a novel cryptographic primitive, based on threshold RSA, that facilitates the secret sharing among multiple nodes in the URSA architecture.

One salient feature of URSA is that it delegates the membership control authority to a group of nodes, without completely trusting any single node. In URSA, each node acts as an access control agent that monitors its local neighbors independently, while multi-node consensus, enabled by secret sharing, is employed in deciding whether to admit or evict a node. This way, no single node monopolizes the system; instead, multiple local nodes monitor each other and jointly make the access control decisions. To the best of our knowledge, URSA is the first work that regulates network memberships in such a democratic peer-to-peer manner.

The benefits of URSA have been highlighted in the literature through its desirable network performance, e.g., scalability to large node population, robustness to wireless channel errors and/or node mobility, and high degree of service availability despite the network dynamics. The basic concepts of URSA are also shown to be applicable in other contexts such as the emerging peer-to-peer networks and applications [6].

While URSA was designed as a practical network security system, rather than a cryptographic theory by any means, there has recently been an increasing interest from the cryptography community to examine the security features of URSA through crypto-analysis [6, 2]. We believe that these efforts are invaluable towards a deep understanding on URSA. In fact, an interesting attack, called the *key search attack*, against URSA has been discovered in [2]. Unfortunately, the authors of [2] misunderstand several critical issues in our original design, and overlook the practical constraints in realistic networking systems. As a result, their analytical results are flawed and misleading. In this article, we are motivated to clarify our design, revisit the crypto-analysis, and shed insight onto both protocol design and network practices.

Our main conclusion drawn from the corrected crypto-analysis is that the key search attack is valid only under strong assumptions, and thus unmountable in practice. The fundamental reason is that this attack requires the adversary to arbitrarily manipulate the secret share update process, which is not allowed in URSA at the first place. Moreover, because the secret shares are updated periodically, to launch the key search attack, the adversary has no choice other than waiting for the scheduled share update operations. That is, the paramount factor that limits the adversary is not the computational capability any more, but the network operational lifetime. For example, even the original analysis in [2] shows that the adversary needs 328 runs of share update to break the system, i.e., almost 6 *months* with a typical setting of two share updates per day. While this is sufficient for most ephemeral ad-hoc networks, our corrected crypto-analysis also validates the security of URSA even in a network that lasts as long as 6 years.

It is important to note that our crypto-analysis, as well as that in [6, 2], applies only to the specific threshold RSA algorithm, *TS-RSA*, described in [4, 5]. We emphasize that the URSA architecture and protocol suite are not bound to any threshold cryptosystems, and their strength in terms of network performance (except than the associated computational overhead) holds when we apply other cryptographic primitives. In fact, the history of security design has seen many examples of such algorithm evolution. For example, in the course of Pretty Good Privacy (PGP), the Bass-O-Matic algorithm used in the initial PGP release was soon replaced with IDEA in PGP 2, and further enhanced by CAST and Triple-DES in PGP 5. Nevertheless, the architecture of PGP remains the same and has been widely accepted in many contexts.

While our analysis demonstrates the security of *TS-RSA* in realistic networking scenarios, we do not claim, by any means, that *TS-RSA* is provably secure or the only choice for the URSA architecture. In fact, alternative URSA implementations [7, 6] based on DSS and BLS signature schemes have recently been reported. However, a crypto-analysis of these schemes or a comparative study is out of scope in this article.

2 On key-search attack on TS-RSA

A recent draft from Jarecki et al. [2] presented an interesting attack against URSA based on TS-RSA, which we term as the *key search attack*. In this attack, the adversary compromises $K-1$ nodes[1], and these compromised nodes do not exhibit any malicious

[1]Recall that K is the critical security parameter in URSA. The global secret key SK is shared by all nodes through a polynomial of degree $K-1$. As a result, a group of K nodes can jointly recover SK or

behavior (e.g., dropping packets or jamming) to avoid being evicted; instead, they properly participate in the network and perform passive attacks to discover the global secret key SK. The key search attack consists of two interleaved subattacks:

- *Attacks against multisignature protocol*: The $K-1$ compromised nodes frequently renew their tickets, and ensure that each new ticket is jointly signed by the $K-1$ compromised nodes and one legitimate node. The multi-signature protocol used for ticket renewal involves the execution of a *k-bounded offsetting algorithm*. By checking the results yielded by this algorithm, the adversary knows whether SK is larger or smaller than a particular value, which is fixed for a given signing group. This information can be used to search the key space. Between two consecutive share updates, the adversary may narrow the search interval by a factor of $K+1$.

- *Attacks against share update protocol*: In a share update operation, if one malicious node happens to reside in the initial K-nodes community, it speaks last in this community to influence the share update polynomial into an arbitrary form that the adversary desires. This way, the adversary can continue the above search in the subsequent time period before the next share update.

The authors of [2] performed a crypto-analysis on the key search attack, and showed negative results on the security of TS-RSA based URSA. While such crypto-analysis is undoubtedly useful in a thorough evaluation of TS-RSA based URSA, we disagree to their conclusion mainly because their analysis is based on several misunderstanding about the design, and overlooks the practical constrains in real networks. A corrected crypto-analysis, as we will present below, actually shows that URSA is secure against the key search attack in practice.

2.1 Periodic Share Update

The analysis in [2] misunderstands the purposes and mechanisms of secret share update in TS-RSA based URSA. The share update protocol was designed to refresh the system and wipe out the secret shares possessed by the evicted malicious nodes, rather than "defending against mobile adversaries" (Section 3.3 in [2]). Instead, such mobile adversaries are handled in URSA by ticket revocation, either implicitly through ticket expiration or explicitly. More important, [2] claims that "The share updates can be both proactive (periodic) as well as reactive, due to revocations of some players from the group" (Sections 2 and 5). This claim is also wrong. *There is no reactive share update in URSA; instead, we only periodically update the shares in the network.*

The above clarification has significant implications on the conclusion that one can draw from the analysis. Even with the original analysis in [2] (which is flawed as we will describe shortly), the adversary needs 328 executions of the share update protocol to succeed. Given that share update is performed infrequently, say twice per day, this simply means that the adversary has to wait for 164 days, i.e., 5.5 months, to break the system. Note that this is already sufficient for most MANETs that are not long-lasting by nature. Even for MANETs with 5.5 months or longer life time, way before the

sign a ticket using SK, but less than K nodes can never do that.

attacker succeeds a rekeying should be scheduled, as most practical security systems do. As such, the sole periodical mode of share updates implies that the dominating factor in evaluating this attack is not the adversary's computational capability as in most theoretical crypto-analysis, but the network operational lifetime as a practical metric.

In the next subsection, we will show that the analysis in [2] is based on yet another misunderstanding about URSA, and the adversary actually needs orders of magnitude more rounds of share updates to break the system than their original results. Nevertheless, as a side note, we mention that the numeric examples used in [2] are misleading because of improper parameter settings. For example, the public exponent e is set as 3 in deriving the results of "157 runs of the update protocol (to break the system)" (Section 5 in [2]). However, in our publications, the recommended value for e is 65537, and all our experimental evaluations are based on this value (e.g., Section 6 in [4]). Needless to say, the security strength of any cryptosystem in practice directly depends on the security parameter settings. For example, even with the Triple-DES algorithm, which is believed as one of the most secure encryption algorithms to date, a 20-bit key still poses severe security threats. However, it is the key setup rather than the Triple-DES algorithm that should take the blame.

2.2 Two-stage Share Update

As described before, the key search attack consists of two interleaved attacks against the multisignature protocol and the share update protocol, respectively. In its original form in [2], the first attack enables the adversary to determine a few bits (3 bits when $K = 7$) in SK between two consecutive share updates, and the second attack allows the adversary to continue to search the key space in an optimal way, by manipulating the share update polynomial into any form that he desires. Thus, an important question arises as follows: *Whether the adversary can completely control the share update polynomial?* Note that the answer is critical in evaluating the effectiveness of key search attack: A negative one simply breaks the original chain of attacks, and the adversary would gain nothing more than the first few bits in SK, which is almost negligible information given that SK can be 1024 or more bits.

In answering the above question, [2] claims that "The attack (against share update protocol) succeeds as long as the adversary corrupts at least one player from the group of players that create the share updating polynomial, and as long as that player can speak last in share update protocol" (Section 4.3). However, this claim is wrong because it misunderstands how the two-stage share update protocol in URSA works. In fact, *the adversary has at most a chance of $\frac{K}{N}$ to completely control the share update polynomial*, where N is the total number of nodes in the network.

For clarification purposes, we briefly overview our two-stage share update protocol as follows. In each periodical update, any K neighboring nodes can form a local community and update the entire network in two stages.

- *Stage 1:* The K nodes jointly generate an update polynomial and update their own shares, using Herzberg's proactive share update algorithm[1].
- *Stage 2:* The share update polynomial, signed by SK, is propagated in the network, so that each and every node gets updated.

There are actually two options for stage 2. The other option is to follow the self-initilization process to update the rest of the network. However, the choice of specific mechanism for stage 2 is not important for our analysis below.

Because such a K-nodes community is formed locally, there exist multiple communities that all initiate the above process. In order to reach network-wide agreement on the share update polynomial, the unique node IDs can be used to arbitrate which community wins over the others. A simple rule, as stated in [3], is a lowest-ID one: the community with the lowest-ID node wins in deciding the share update polynomial.

Now we can see why the claim about the share update attack in [2] is wrong. In the scenario that it argues, the adversary controls $K-1$ nodes, while one of them resides in a K-nodes community and speaks last in the community. Let Γ denote the group of $K-1$ compromised nodes, Ω denote the K-nodes community, and B denote the single compromised node in Ω. Note that B has to announce its partial polynomial in Stage 1. However, in this stage, the adversary knows only one sample point on each of the $K-1$ partial polynomials[2] contributed by $\Omega\backslash\{B\}$, because the $K-2$ compromised nodes in $\Gamma\backslash\{B\}$ have not received any polynomial yet. Therefore, even if B speaks last in Ω, the adversary can only set new shares for set B arbitrarily, but he can never predict or decide the new shares of the $K-2$ compromised nodes in $\Gamma\backslash\{B\}$. This shows that the share update attack, in particular the optimal setting of new shares for the compromised nodes, is invalid as presented in [2].

2.2.1 Corrected Crypto-analysis

Given that the original attack described in [2] is invalid, we now go one step further to see whether it can revised into a sound attack. While a full exploration of this issue remains an open problem, we provide below one such revision as our initial efforts.

In the revised attack, the adversary has to put all $K-1$ compromised nodes in one local neighborhood, so that they can form a K-nodes group with one another legitimate node. Only based on the $K-1$ samples obtained by those $K-1$ compromised nodes in the group, the adversary can recover the update polynomial contributed by the legitimate node. In this scenario, the attacker can have one compromised node to speak last, so arbitrate the aggregated polynomial to set the new shares of all $K-1$ compromised nodes.

However, even with the revised attack, the adversary cannot succeed all the time. This is because the adversary can control at most one group with $K-1$ compromised nodes. Meanwhile, multiple K-nodes communities co-exist in the network and compete with each other in deciding the ultimate share update polynomial. With the lowest-ID rule, the adversary succeeds only if the node with the lowest ID happens to reside in the community under his control. When the adversary cannot selectively compromise nodes, i.e., corrupting any $K-1$ nodes as he wishes, the chance of a successful share update attack is roughly $\frac{K}{N}$, in which N is the total number of nodes in the network.

Even if the adversary is able to selectively compromise nodes, the above result still holds with very simple design extensions such as a lowest-ID-hash rule. In this rule, we apply a secure hash function (e.g., MD5) on the node IDs, and the community with the

[2] A share update polynomial has a degree of $K-1$, but the constant coefficient is zero. Therefore, to recover this polynomial, the adversary needs to know at least $K-2$ samples.

lowest-ID-hash node wins over the others. This way, the adversary cannot derive which node he should compromise to guarantee the success of share update attacks.

With the corrected crypto-analysis, the adversary needs significantly more runs of share update to break the system than the original results in [2]. Specifically, the number of required runs increases by at least a factor of $\frac{N}{K}$. This simply implies that TS-RSA based URSA can securely operate for a much longer period of time. For example, in a typical setting of $K = 7$ and $N = 100$, the allowed network operational lifetime increases by at least a factor of 14. Given the original result of 5.5 months to break the system in Section 2.1, the corrected crypto-analysis actually shows that the network can continuously operate for 77 months, i.e., more than 6 years. We believe that this is long enough in practice before the global public/secret key pair is reset, say once per year.

3 Conclusion

To this end, we have clarified several misunderstandings in the recent crypto-analysis on URSA, and demonstrated the security of our *TS-RSA* algorithm in the realistic networking scenarios. As argued by Bruce Schneier, "In theory there is no difference between theory and practice. In practice there is."[8]. We believe that the network security research should benefit from the collaborative efforts from both cryptograph and network communities. Yet we also believe that in designing and analyzing a network security system, it is critical to understand the practicality and applicability of theoretical results. Put aside the network and system performance issues which can be dominant in practice, the lessons provided in this article certainly is one example that a theoretically strong attack becomes almost invalid in practice with careful protocol designs.

References

[1] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive Secret Sharing or: How to Cope with Perpetual Leakage. extended abstract, IBM T.J. Watson Research Center, November 1995.

[2] S. Jarecki, N. Saxena, and J. H. Yi. Cryptanalysis of the Proactive RSA Signature Scheme in the URSA Ad-Hoc Network Access Control Protocol. In submssion, 2004.

[3] J. Kong, P. Zerfos, H. Luo, S. Lu, and L. Zhang. Providing Robust and Ubiquitous Security Support for Mobile Ad-hoc Networks. In *IEEE ICNP'01*, pages 251–260, 2001.

[4] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang. URSA: Ubiquitous and Robust Access Control for Mobile Ad Hoc Networks. *IEEE/ACM Transactions on Networking*, to appear.

[5] H. Luo and S. Lu. Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks. Technical Report TR-200030, Dept. of Computer Science, UCLA, 2000.

[6] N. Saxena, G. Tsudik, and J. Yi. Admission Control in Peer-to-Peer: Design and Performance Evaluation. In *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 104–114, 2003.

[7] N. Saxena, G. Tsudik, and J. Yi. Group Memberships as a Resource: Decentralized Admission and Eviction. In submission, 2004.

[8] B. Schneier. *Secret and Lies, Digital Security in a Networked World*. Wiley Computer Publishing, 2000.