Problem Set 5 Solutions

Problem 5-1. SPKI/SDSI

Following is the solution submitted by Ryan Barrows, Alex DeNeui, Atish Nigam.

This modification to SPKI/SDSI is an interesting and useful model, however, there are a few serious flaws that make this particular modification not meet the goals of its intention. The timestamp feature of the certificate allows a user to create a new key pair and invalidate all previous key pairs and as a result invalidate all previous public keys. From a usefulness perspective, this modification is a great benefit because it allows a user to replace his public/private key pair whenever he feels it necessary. Furthermore, from an ease-of-use perspective, allowing each client to replace their public key by changing a reducing name cert on their computer is very easy to use and implement. It only involves the client and a single certificate authority.

However, there is a serious flaw in the security of this modified system. In the SPKI/SDSI model, a protected resource does not need to have any information about individual clients or their certificate chains. A protected resource simply releases information to a client if that client can prove a valid certificate chain. In this modified system, the problem lies in the fact that a protected resource has no mechanism for knowing when and if a reducing name cert, $KA \rightarrow K'$, is valid or has been invalidated and replaced by a newer certificate. Thus, the security of the modified system is severely hindered. Say for instance a clients reducing name cert, $Kclient \rightarrow K$, is compromised to an eavesdropper. This client may then create a new cert, $Kclient \rightarrow K'$, and think that the old one is now invalidated. However, since all the protected resources he is accessing do not realize the difference between the valid and invalid certificate, the eavesdropper will still be able to access all the client's protected resources since the eavesdropper will be able to produce a certificate chain of signed certificates that map to the necessary root certificate.

Should this modification be implemented there would need to be a change to the design to make it workable. One such change could be that a central repository (a web page, newsletter, etc) would list each clients' most recent reducing name cert timestamp. Then each protected resource could access this repository and verify that a user was showing the most recent and valid certificate. Without such certificate timestamp reporting, the modification would not be applicable. Unfortunately, with this timestamp reporting in place, the system would then become unreasonable. The primary goal of the SPKI/SDSI system is that it eliminates the need for a central repository of certificates or public keys and places this responsibility on the individual clients. A central repository of timestamps would go against this idea and would invalidate the SPKI/SDSI implementation.

Collaboration

Atish wrote this problem after a group discussion.

Problem 5-2. Term projects

We really enjoyed reading your project proposals, and now updates. Should you have any questions or problems, don't hesitate to email us!!!

Problem 5-3. Zero knowledge

Recall the clarification sent by email regarding how the Verifier checks the validity of the 3^{rd} message.

- If $b = 0$, the Verifier receives the pair (r', r''). He computes
  - $(g^0 \cdot r'^N \bmod N^2,\ g^1 \cdot r''^N \bmod N^2)$
  - $(g^1 \cdot r'^N \bmod N^2,\ g^0 \cdot r''^N \bmod N^2)$

  and accepts if either of these pairs matches the first message $(c',\ c'')$.
- If $b = 1$, the Verifier receives a single value z. He computes $z^N \bmod N^2$ and accepts if it matches either c/c' or c/c''.

(a) Completeness

Solution Note: The purpose of this question is to *actually show* that the way the Verifier checks the proof is correct, i.e. that the Verifier accepts if c does encode $m = 0$ or $m = 1$ and the Prover follows the protocol.

We show by cases that, if c encodes $m = 0$ or $m = 1$, the Prover always succeeds.

- $b = 0$, $m' = 0$.
  The Prover sends the pair $(c',\ c'') = (g^0 \cdot r'^N \bmod N^2,\ g^1 \cdot r''^N \bmod N^2)$, which matches the first pair computed by the Verifier, upon reception of the pair $(r',\ r'')$.
- $b = 0$, $m' = 1$.
  The Prover sends the pair $(c',\ c'') = (g^1 \cdot r'^N \bmod N^2,\ g^0 \cdot r''^N \bmod N^2)$, which matches the second pair computed by the Verifier, upon reception of the pair $(r',\ r'')$.
- $b = 0$, $m' = m$.

$$\begin{aligned}\frac{c}{c'} &= \frac{g^m \cdot r^N \bmod N^2}{g^m \cdot r'^N \bmod N^2} \\ &= \frac{r^N \bmod N^2}{r'^N \bmod N^2} \\ &= \frac{r^N}{r'^N} \bmod N^2 \\ &= \left(\frac{r}{r'}\right)^N \bmod N^2\end{aligned}$$

  The Prover sends the value $z = r/r' \bmod N^2$, which will be such that $z^N \bmod N^2 = c/c'$, and make the Verifier accept.
- $b = 0$, $m' = 1 - m$.

$$\begin{aligned}\frac{c}{c''} &= \frac{g^m \cdot r^N \bmod N^2}{g^m \cdot r''^N \bmod N^2} \\ &= \frac{r^N \bmod N^2}{r''^N \bmod N^2} \\ &= \frac{r^N}{r''^N} \bmod N^2 \\ &= \left(\frac{r}{r''}\right)^N \bmod N^2\end{aligned}$$

The Prover sends the value $z = r/r'' \bmod N^2$, which will be such that $z^N \bmod N^2 = c/c''$, and make the Verifier accept.

(b) Soundness

Solution Note: Many groups proved that the two obvious strategies to cheat the Verifier — picking (c', c'') according to the protocol to "cheat" the case $b = 0$, and picking (c', c'') so as to know the N^{th} root asked for to cheat the case $b = 1$ — are unsuccessful.
You have to prove that the Prover cannot cheat using *any* strategy. The core of that proof essentially comes down to the two above-mentioned strategies as we will see, but the difference in the two approaches, though subtle, is essential.

If c does not encode $m = 0$ or $m = 1$, we first prove that he cannot succeed in *one* round of the protocol with probability more than $\frac{1}{2} + \epsilon$.

- If the pair (c', c'') is not constructed in the way the protocol specifies it, then there is negligible probability ϵ_1[1] that it will match one of the two pairs computed by the Verifier — $(g^0 \cdot r'^N \bmod N^2, g^1 \cdot r''^N \bmod N^2)$ or $(g^1 \cdot r'^N \bmod N^2, g^0 \cdot r''^N \bmod N^2)$ — in the case where he sends the challenge $b = 0$. Thus the Prover will succeed with at most probability ϵ_1 if $b = 0$ (and at most probability 1 if $b = 1$).
  The Prover will therefore succeed with at most probability $\frac{1}{2} \cdot \epsilon_1 + \frac{1}{2} \cdot 1$ in our first case.
- If the pair (c', c'') is constructed in the way the protocol specifies it (in particular $m' \in \{0, 1\}$), then the Prover will succeed with probability 1 if $b = 0$.
  But then if $b = 1$, the Prover needs to provide an N^{th} root mod N^2 for either $c/c' = g^{m-m'} \cdot (\frac{r}{r'})^N$ or $c/c'' = g^{m-1+m'} \cdot (\frac{r}{r''})^N$. This means that he has to be able to compute an N^{th} root mod N^2 for either $g^{m-m'}$ or $g^{m-1+m'}$, that is for either g^m or g^{m-1}, with $m \neq 0$ and $m \neq 1$ (since $m' \in \{0, 1\}$).
  Now, as the Prover does *not* know $\varphi(N^2) = N \cdot \varphi(N)$, he cannot compute non-trivial N^{th} roots mod N^2. So that there is negligible probability ϵ_2[1]that he can actually provide an N^{th} root mod N^2 for either c/c' or c/c''.
  The Prover will therefore succeed with probability $\frac{1}{2} + \frac{1}{2} \cdot \epsilon_2$ in our second case.

In any case, the Prover will succeed in *one* round of the protocol with at most probability $\frac{1}{2} + \epsilon \approx \frac{1}{2}$ (where $\epsilon = \max(\frac{\epsilon_1}{2}, \frac{\epsilon_2}{2})$).

In conclusion, if c does not encode $m = 0$ or $m = 1$, the Prover will succeed in the protocol with at most probability $(\frac{1}{2} + \epsilon)^t \approx \frac{1}{2^t}$. He will thus be caught with at least probability $1 - \frac{1}{2^t} \overset{t \to \infty}{\longrightarrow} 1$.

(c) Zero-knowledge

Solution Note: You are to prove the zero-knowledge property by explicitly showing a simulation algorithm. This formal setting has been introduced to avoid vague, hand-waving proofs that the protocol does not leak any knowledge, but the property to be proven.

Following is one algorithm simulating the protocol:

[1] The Prover cannot do better than random guessing.

1. Pick $b \in_R \{0,1\}$.
2. If $b = 0$:
   (a) Pick $m' \in_R \{0,1\}$, $r' \in_R \mathbb{Z}_N^*$, and $r'' \in_R \mathbb{Z}_N^*$.
   (b) Compute $c' = g^{m'} \cdot r'^N \bmod N^2$ and $c'' = g^{1-m'} \cdot r''^N \bmod N^2$.
   (c) Output the transcript:

1^{st} **message:** (c', c'')
2^{nd} **message:** 0
3^{rd} **message:** (r', r'')

   If $b = 1$:
   (a) Pick $z \in_R \mathbb{Z}_{N^2}^*$.
   (b) Compute $t = z^N \bmod N^2$.
   (c) Pick $\rho \in_R \mathbb{Z}_N^*$, and compute $E(1) = g^1 \cdot \rho^N \bmod N^2$.
   (d) Pick $b' \in_R \{0,1\}$.
   If $b' = 0$: Compute $c' = c/t \bmod N^2$, and $c'' = E(1)/c' \bmod N^2$.
   If $b' = 1$: Compute $c'' = c/t \bmod N^2$, and $c' = E(1)/c'' \bmod N^2$.
   (e) Output the transcript:

1^{st} **message:** (c', c'')
2^{nd} **message:** 1
3^{rd} **message:** z

In both the actual protocol and the simulation, half of the messages have $b = 0$ and half have $b = 1$.

- The messages with $b = 0$ will have the same distribution in both the actual protocol and the simulation, since in the simulation, the Verifier essentially proceeds as the Prover would (and does not need to know anything about the ciphertext c to do so).
- The harder case is the distribution of the messages with $b = 1$.
  Since $m' \in_R \{0,1\}$, we will have $m' = m$ half of the time, and $m' = 1 - m$ the other half in the actual protocol, which means the Prover reveals the N^{th} root of c/c' half of the time and of c/c'' the other half.
  In the simulation, in the case $b' = 0$ (which occurs half of the time), the 3^{rd} message z is such that $z^N \bmod N^2 = t = c/c' \bmod N^2$, i.e. is an N^{th} root of c/c'; likewise, in the case $b' = 1$ (which occurs the other half of the time), the 3^{rd} message z is an N^{th} root of c/c''.
  Finally, as in the actual protocol $c' \cdot c'' = E(m') \cdot E(1 - m') = E(m' + 1 - m') = E(1)$, we made sure that in the simulation we also have $c' \cdot c'' = E(1)$.

In conclusion, both the actual protocol and the simulation have the same distributions of transcripts.