Parameterized Verification of Timed Security Protocols with Clock drift

Li Li, Jun Sun and Jin Song Dong
Motivation

“Since clock synchronization is so important in the security of the Kerberos protocol, if clocks are not synchronized within a reasonable window Kerberos will report fatal errors and refuse to function.”

It is advisable to set Maximum tolerance for computer clock synchronization to a value of 5 minutes.

What kind of clock drifts are safe?
How do we formally answer such questions?
We are responsible for answering the questions!
Research Questions

How do we model timed security protocols?

How do we model clock drifts?

How do we verify the models?
A Running Example

Corrected Wide Mouthed Frog (WMF)

- a key exchange protocol
- verified to be secure assuming clocks are perfectly synchronized

Alice

Bob

Server
Corrected WMF

1. send \(<t_a, B, k, tag1>\) encrypted using key(A)

2. receive at \(t_s\)
check \(t_s - t_a \leq p\)

3. send \(<t_s, A, k, tag2>\) encrypted using key(B)

4. receive at \(t_b\)
check \(t_b - t_s \leq p\)
accepts session key \(k\)
Modeling Corrected WMF

Timed Applied π-Calculus

1. send \( \langle t_a, B, k, \text{tag1} \rangle \) encrypted using key(A)

\[
P_a \triangleq \text{in}(r) . \nu k . \mu t_a : c_a . \text{init}(A, r, k) @ t_a . \overline{\text{out}}(\langle A, \text{enc}_s(\langle t_a, r, k, \text{tag1} \rangle, \text{key}(A)) \rangle).0
\]
Modeling Corrected WMF

Timed Applied π-Calculus

4. receive at $t_b$
   check $t_b - t_s \leq p$
   accepts session key $k$

$$P_b \triangleq c(x).\mu t_b : c_b.\text{let } \langle t_s, =A, k, =\text{tag}_2 \rangle = \text{dec}_s(x, \text{key}(B)) \text{ then }$$

   if $t_b - t_s \leq p_m$ then check $k$ in db as unique then accept$(A, B, k)@t_b.0$
Modeling Corrected WMF

\[ P \triangleq (\neg P_r) \neg (\neg P_a) \neg (\neg P_s) \neg (\neg P_b) \neg (\neg P_p) \]
Timed Logic Rules

\[
[ G ] e_1, e_2, \ldots, e_n \quad \neg [ B ] \rightarrow e
\]

G: an untimed guard condition; e: an event; B: a timed constraint

Rules from the protocol model

Rules modeling the attacker
Model Rules

\[ \text{know}(t_b, t_b), \text{know}(\text{enc}_s(\langle t_s, A[], k, \text{tag}_2[] \rangle, \text{key}(B[])), t_1) \]

\[ \neg [t_1 \leq t_b \land t_b - t_s \leq \$p_m] \rightarrow \text{accept}([n_b], \langle A[], B[], k \rangle, t_b) \]

Assume no clock drift now

4. receive at \( t_b \)
check \( t_b - t_s \leq p \)
accepts session key \( k \)
Attacker Model

Delov-Yao Attacker Model, e.g.

\[
know(m, t_1), know(k, t_2) \rightarrow [ t_1 \leq t \land t_2 \leq t ] \rightarrow know(\text{enc}_s(m, k), t)
\]

More than Delov-Yao, e.g.

\[
know(\text{RC4}(m, k), t_1) \rightarrow [ t - t_1 > \frac{1}{3}d ] \rightarrow know(k, t)
\]
Modeling Clock Drift

**VR** (Variable Rate):

Different clocks have different clock rates and there is a maximum bound on the drift

**SR** (Same Rate):

Different clocks share the clock rate but have different readings
Clock Drift: VR

\[
know(t_b, t_b), \, know(\text{enc}_s(\langle t_s, A[], k, tag_2[]\rangle, \text{key}(B[])), t_1) \\
-\left[ t_1 \leq t_b \land t_b - t_s \leq \$p_m \right] \rightarrow accept([n_b], \langle A[], B[], k\rangle, t_b)
\]

\[
know(t_b, t_b'), \, know(\text{enc}_s(\langle t_s, A[], k, tag_2[]\rangle, \text{key}(B[])), t_1) \\
-\left[ t_1 \leq t_b' \land t_b - t_s \leq \$p_m \land |t_b' - t_b| \leq \$p_b \right] \rightarrow accept([n_b], \langle A[], B[], k\rangle, t_b')
\]
Clock Drift: SR

\[
\begin{align*}
\text{know}(t_b, t_b), \text{know}(\text{enc}_s(\langle t_s, A[], k, tag_2[] \rangle, \text{key}(B[])), t_1) \\
- [ t_1 \leq t_b \land t_b - t_s \leq \delta p_m ] \rightarrow \text{accept}([n_b], \langle A[], B[], k \rangle, t_b)
\end{align*}
\]

\[
\begin{align*}
\text{know}(t_b, t_b'), \text{know}(\text{enc}_s(\langle t_s, A[], k, tag_2[] \rangle, \text{key}(B[])), t_1) \\
- [ t_1 \leq t_b' \land t_b - t_s \leq \delta p_m \land t_b - t_b' = \delta d_b ] \rightarrow \text{accept}([n_b], \langle A[], B[], k \rangle, t_b')
\end{align*}
\]
Research Questions

How do we model timed security protocols?

How do we model clock drifts?

How do we verify the models?
Verification: Property

Non-injective timed authentication

For every acceptance of the protocol responder, the protocol initiator indeed initiates the protocol the protocol and protocol partners indeed join in the protocol, agreeing on the protocol arguments and timing requirements.

\[
accept(i, r, k)@t_r \leftrightarrow [\ t_s - t_i \leq \delta p_m \wedge t_r - t_s \leq \delta p_m ] \rightarrow init(i, r, k)@t_i, join(i, r, k)@t_s
\]

Another rule.
Verification Algorithm

Take two rules to generate a new rule;

If the new rule is not subsumed by any existing rule, add the new rule.

If the events in one of the rules match those of the property (init, join, accept), output the time constraint as the verification result.

Rules are abstracted for termination.
Rule Composition

\[ \text{know}(m, t_1), \text{know}(k, t_2) \rightarrow [t_1 \leq t \land t_2 \leq t] \rightarrow \text{know}(\text{enc}_s(m, k), t) \]

+ 

\[ \text{know}(t_b, t_b), \text{know}(\text{enc}_s(\langle t_s, A[]\rangle, k, \text{tag}_2[]), \text{key}(B[])), t_1) \rightarrow [t_1 \leq t_b \land t_b - t_s \leq \delta_p m] \rightarrow \text{accept}([n_b], \langle A[], B[], k\rangle, t_b) \]

\[ \text{know}(t_b, t_b), \text{know}(\langle t_s, A[]\rangle, k, \text{tag}_2[]), t_1), \text{know}(\text{key}(B[]), t_2) \rightarrow [t_1 \leq t_b \land t_2 \leq t_b \land t_b - t_s \leq \delta_p m] \rightarrow \text{accept}([n_b], \langle A[], B[], k\rangle, t_b) \]
## Evaluation

<table>
<thead>
<tr>
<th>Protocol</th>
<th>( #R )</th>
<th>No Clock Drift</th>
<th>Shared Clock Rate</th>
<th>Variable Clock Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Result</td>
<td>Time</td>
<td>Result</td>
<td>Time</td>
</tr>
<tr>
<td>Corrected WMF [7,18,16]</td>
<td>80</td>
<td>Secure 47.51ms</td>
<td>Threat 112.75ms</td>
<td>Attack 150.09ms</td>
</tr>
<tr>
<td>TESLA [22,21]</td>
<td>343</td>
<td>Secure 3.17s</td>
<td>Threat 3.55s</td>
<td>Threat 4.37s</td>
</tr>
<tr>
<td>Auth Range [6,8]</td>
<td>53</td>
<td>Secure 38.58ms</td>
<td>Secure 60.73ms</td>
<td>Attack 46.47ms</td>
</tr>
<tr>
<td>CCITT X.509 (1c) [3]</td>
<td>135</td>
<td>Secure 162.69ms</td>
<td>Secure 231.86ms</td>
<td>Secure 224.00ms</td>
</tr>
<tr>
<td>CCITT X.509 (3) BAN [7]</td>
<td>198</td>
<td>Secure 791.00ms</td>
<td>Secure 1058.05ms</td>
<td>Secure 969.97ms</td>
</tr>
<tr>
<td>NS PK Time [20,17,10]</td>
<td>173</td>
<td>Secure 170.00ms</td>
<td>Threat 205.93ms</td>
<td>Threat 353.20ms</td>
</tr>
</tbody>
</table>

Secure: some trivial time constraint has to be satisfied
Threat: some nontrivial constraint has to be satisfied
Attack: there is always an attack
Case Study: TELSA

Designed with clock drifts

No clock drift or Shared Clock Rates:

  Verification Result: 2*network latency < interval

Variable Clocks:

  Verification Result: drift_s + drift_r <= interval
Conclusion

We have developed a tool to verify security protocols with clock drifts.

This line of work is based on ProVerif.

Ongoing Work

“Since clock synchronization is so important in the security of the Kerberos protocol, if clocks are not synchronized within a reasonable window Kerberos will report fatal errors and refuse to function.”

It is advisable to set Maximum tolerance for computer clock synchronization to a value of 5 minutes.

Unfortunately, the current implementation is not efficient enough to verify Kerberos V once clock drift is considered.