How to Store Some Secrets

Reto E. Koenig, Rolf Haenni

University of Fribourg
&
Bern University of Applied Sciences

12.07.2011
Usability Studies JCJ-05

The voter has to memorize different credentials with very high entropy:

Real  The credential for the real voting act
Fake  The credential used to deceive the adversary
Question

How to store and discriminate these credentials without hinting the adversary?
Hardening JCJ-05 for reality

**Speedup**  JCJ-05 is too slow for large scale elections

**Board flooding**  Easy to bring down JCJ-05 by a denial of service attack.

Mission accomplished, Problems solved. \[KHS11\]

---

Preventing Board Flooding Attacks in Coercion-Resistant Electronic Voting Schemes  
SEC'11, 26th IFIP International Information Security
Hardening JCJ-05 for reality

**Speedup**  JCJ-05 is too slow for large scale elections

**Board flooding**  Easy to bring down JCJ-05 by a denial of service attack.

Mission accomplished, Problems solved.\[^{KHS11}\]

\[^{KHS11}\]  R. Koenig and R. Haenni and S. Fischli
Preventing Board Flooding Attacks in Coercion-Resistant Electronic Voting Schemes

SEC’11, 26th IFIP International Information Security
Usability Studies on the Hardened JCJ05 Derivate

Each voter...

• ...needs to secretly store several dozens credentials
• ...has to discriminate doubtless between credentials for ’Accept’ and ’Fake’.
• ...is not allowed to mark any credential
• ...shall never unveil the amount of possessed secrets (They vary per voter)
Usability Studies on the Hardened JCJ05 Derivate

Each voter...

- ...needs to secretly store several dozens credentials
- ...has to discriminate doubtless between credentials for 'Accept' and 'Fake'.
- ...is not allowed to mark any credential
- ...shall never unveil the amount of possessed secrets (They vary per voter)
Usability Studies on the Hardened JCJ05 Derivate

Each voter...

- needs to secretly store several dozens credentials
- has to discriminate doubtless between credentials for 'Accept' and 'Fake'.
- is not allowed to mark any credential
- shall never unveil the amount of possessed secrets (They vary per voter)
Question

How to manage...
Problem Domain

Strategies

- **Password vault with a single master password**
  - Challengeable 'offline'
  - Once open, every credential visible

- **One ciphertext per credential**
  - Managing ciphers
  - Match password and cipher... Which is what?

- **Secret-Storage System**
  - Well...
Problem Domain Secret Storing

Strategies

- Password vault with a single master password
  - Challengeable 'offline'
  - Once open, every credential visible
- One ciphertext per credential
  - Managing ciphers
  - Match password and cipher... Which is what?
- Secret-Storage System
  - Well...
Strategies

- Password vault with a single master password
  - Challengeable 'offline'
  - Once open, every credential visible
- One ciphertext per credential
  - Managing ciphers
  - Match password and cipher... Which is what?
- Secret-Storage System
  - Well...
Strategies

- **Password vault with a single master password**
  - Challengeable 'offline'
  - Once open, every credential visible

- **One ciphertext per credential**
  - Managing ciphers
  - Match password and cipher... Which is what?

- **Secret-Storage System**
  - Well...
Properties of a Secret-Storage System

The system...

- ...allows to choose freely $n$ keys
- ...allows to choose freely $n$ secrets
- ...allows to store multiple secrets in one storage (aka cipher)
- ...allows to retrieve only the secret correlated to the key
- ...has all properties of a (symmetric) crypto-system
Definition of a Secret-Storage System

\[ \Sigma[n] = (S, K, C, \text{store, retrieve}) \]
Problem Domain: Secret Storing

\[ \Sigma[n] = (S, K, C, \text{store}, \text{retrieve}) \]

- **S** = *secret space*, set of all possible secrets
- **K** = *key space*, set of all possible keys
- **C** = *storage space*, the set of all possible storages

**store**: \( S^n \times K^{(n)} \rightarrow C \)

*storage function*, where \( K^{(n)} \subseteq K^n \) is the set of all admissible key tuples (with distinct keys)

**retrieve**: \( C \times K \rightarrow S \)

*the retrieval function*

Reto E. Koenig, Rolf Haenni Secret-Storage
\[ \Sigma[n] = (S, K, C, \text{store}, \text{retrieve}) \]

- \( S \) = *secret space*, set of all possible secrets
- \( K \) = *key space*, set of all possible keys
- \( C \) = *storage space*, the set of all possible storages

store : \( S^n \times K^{(n)} \longrightarrow C \)

*storage function*, where \( K^{(n)} \subseteq K^n \) is the set of all admissible key tuples (with distinct keys)

retrieve : \( C \times K \longrightarrow S \)

the *retrieval function*
\[ S = (s_1, \ldots, s_n) \in S^n, \text{ an } n\text{-tuple of secrets } (n \geq 1) \]
\[ K = (k_1, \ldots, k_n) \in K^{(n)}, \text{ an } n\text{-tuple of distinct keys } n \geq 1 \]
\[ c = \text{a particular storage} \]

\[
\text{store}_K(S) = c \in C, \text{ storing the } n\text{-tuple of the secrets } S \in S^n \\
\text{with the } n\text{-tuple of distinct keys } K \in K^{(n)}
\]

\[
\text{retrieve}_{k_i}(c) = s_i \in S \text{ retrieval with key } k_i
\]

\[
\text{retrieve}_{k_i}(\text{store}_K(S)) = s_i
\]
\[ S = (s_1, \ldots, s_n) \in \mathcal{S}^n, \text{ an n-tuple of secrets } (n \geq 1) \]
\[ K = (k_1, \ldots, k_n) \in \mathcal{K}^{(n)}, \text{ an n-tuple of distinct keys } n \geq 1 \]
\[ c = \text{a particular storage} \]
\[ \text{store}_K(S) = c \in \mathcal{C}, \text{ storing the n-tuple of the secrets } S \in \mathcal{S}^n \]
\[ \text{with the n-tuple of distinct keys } K \in \mathcal{K}^{(n)} \]
\[ \text{retrieve}_{k_i}(c) = s_i \in \mathcal{S} \text{ retrieval with key } k_i \]

\[ \text{retrieve}_{k_i}(\text{store}_K(S)) = s_i \]
$S = (s_1, \ldots, s_n) \in S^n$, an n-tuple of secrets ($n \geq 1$)

$K = (k_1, \ldots, k_n) \in K^{(n)}$, an n-tuple of distinct keys $n \geq 1$

$c = \text{a particular storage}$

\[ \text{store}_K(S) = c \in C, \text{ storing the n-tuple of the secrets } S \in S^n \]
\[ \text{with the n-tuple of distinct keys } K \in K^{(n)} \]

\[ \text{retrieve}_{k_i}(c) = s_i \in S \text{ retrieval with key } k_i \]

\[ \text{retrieve}_{k_i}(\text{store}_K(S)) = s_i \]
Properties of the Secret-Storing System

Required to possess the cryptographic properties of a conventional symmetric crypto-system:
- Retrieving $s_i$ from $c$ does not disclose any information about the other secrets in $c$
- Applying $K$ on $c$ returns $S$
- Serves a conditional entropy $H(S|c)$ which is equal to $H(S)$
- Applying $K'$ on $c$ where $K' \neq K$ does return $S$ with a probability of $\frac{1}{|S|}$
Realisation using a Prime Field $\mathbb{Z}_p$, where $p = 7$, $n = 3$

$\mathbb{Z}_p = \{0, 1, 2, 3, 4, 5, 6\}$
Realisation using a Prime Field $\mathbb{Z}_p$, where $p = 7$, $n = 3$

$\mathbb{Z}_p = \{0, 1, 2, 3, 4, 5, 6\}$

$\mathcal{K} = \mathbb{Z}$
Realisation using a Prime Field $\mathbb{Z}_p$, where $p = 7$, $n = 3$

$\mathbb{Z}_p = \{0, 1, 2, 3, 4, 5, 6\}$

$\mathcal{K} = \mathbb{Z}$

$\mathcal{S} = \{0, 1\}$
The \textit{store}-Function in $\mathbb{Z}_p$, where $p = 7, n = 3$

$\mathbb{Z}_p = \{0, 1, 2, 3, 4, 5, 6\}$

$\mathcal{K} = \mathbb{Z}$

$\mathcal{S} = \{0, 1\}$

$\mathcal{S} = (1, 0, 1)$

$\mathcal{K} = (99, 13, 42)$
The *store*-Function in $\mathbb{Z}_p$, where $p = 7, n = 3$

$\mathbb{Z}_p = \{0, 1, 2, 3, 4, 5, 6\}$

$K = \mathbb{Z}$

$S = \{0, 1\}$

$S = (1, 0, 1)$

$K = (99, 13, 42)$

$c = (3, 3, 2)$

$f(x) = 3 + 3x + 2x^2$
The retrieve-Function for the key 99 in $\mathbb{Z}_p$, where $p = 7, n = 3$

$\mathbb{Z} \mapsto \mathbb{Z}_p \quad \kappa(99) = 4$

$f(x) \quad f(4) = 5$

$\mathbb{Z}_p \mapsto S \quad \sigma(5) = 1$

$\mathbb{Z}_p$ ↦ $\mathbb{Z}_p$

$f(x)$

$\sigma(5) = 1$

$\mathbb{Z}_p$ ↦ $S$

$\sigma = 1$

$f(4) = 5$

$\kappa(99) = 4$

$f(x) = 3 + 3x + 2x^2$

$\mathbb{Z}_p \mapsto S$

$\sigma = 1$

$\kappa(99) = 4$

$\mathbb{Z}_p$ ↦ $\mathbb{Z}_p$

$f(x) = 3 + 3x + 2x^2$

$\mathbb{Z}_p$ ↦ $S$

$\sigma = 1$
Now What?

My Credentials for E-Voting