Does Counting Still Count?
Revisiting the Security of Counting based User Authentication Protocols against Statistical Attacks

Hassan Jameel Asghar\textsuperscript{1}, Shujun Li\textsuperscript{2}
Ron Steinfeld\textsuperscript{3}, Josef Pieprzyk\textsuperscript{1}

\textsuperscript{1}Macquarie University, Australia
\textsuperscript{2}University of Surrey, UK
\textsuperscript{3}Monash University, Australia

hassan.asghar@mq.edu.au, shujun.li@surrey.ac.uk
ron.steinfeld@monash.edu, josef.pieprzyk@mq.edu.au
Outline

- An Old Problem: Unassisted Human Authentication against Observers (1990s-)
- Our Contributions
  - Why does Yan et al.’s attack work? – A general theoretical analysis of \( \delta D \) statistical attacks (\( \delta \geq 1 \)) on counting based protocols
  - An approach for estimating the security bound
  - New principles and fixes to make counting based protocols more secure against the new attacks
The (old) problem

- How to authenticate an **unassisted** human user on an **observable (untrusted)** terminal?
  - Why **unassisted**? – Hardware devices cause usability problems and may be attacked as well.
  - Who are **observers**? – Shoulder surfers, hidden cameras, card skimmers, malware (keyloggers, screen scrapers, Trojan horses, MitM/B, etc.)

This problem was first modelled by Matsumoto and Imai (EUROCRYPT’91)
- Challenge-response protocols proposed as general solutions to hide the shared secret $P$ in challenges $C = f_C(P)$ and responses $R = f_R(P, C)$.

- Many solutions exist, but the main research question remains **unanswered**:
  - How to make a protocol which is both usable and secure against adversary with **many** observed sessions?
Solutions based on counting?

- Many proposed solutions follow this approach.
- Password $P = k$ pass-objects out of $n$ objects
- Challenge $C = I$ objects ($I \leq n$)
- Response $R$
  - Count pass-objects $P$ in $C \Rightarrow \#C(P)$
  - Response $R = f_R(\#C(P))$, e.g. $R = \#C(P) \mod 2$
- Why counting?
  - Recognizing objects and counting are believed easy tasks for most human users!
Foxtail: A typical counting based protocol

- Proposed by Li & Shum in 2001/2002 (published as IACR ePrint 2005/268)
- Claimed to be secure: given $O(n)$ observed sessions, the adversary’s chance of success is $2^{-n}$.
- Usability is better than other solutions with similar security, but still not practical (2-3 minutes).
- At NDSS 2012 Yan et al. reported a statistical attack which can fully recover $P$ with $O(n)$ observed sessions.
  - The attack can be generalized to other counting based protocols.
How does Foxtail work?

- Challenge $C$ of size $2l = C_1 + C_2$ (each of size $l$)
  - Uni-Rule: $C_1$ is generated such that there are 0, 1, 2 or 3 pass-objects with equal probability.
  - Rand-Rule: $C_2$ is generated at random (the number of pass-objects can be anything from 0 to $\min(k,l)$).

- Response $R$
  - $R=0$ if $\#C(P)$ mod 4 = 0 or 1, otherwise $R=1$

- Example
  - For the above challenge $C$, the response $R=0$. 
How does Yan et al.’s attack work?

- Based on counting as well (but in 2D space)!
  - For Response 0 and 1, count the occurrences of each object pair \((o_1, o_2)\) in each challenge to get \(F_1\) and \(F_2\).
  - Rank all objects pairs according to \(F_1-F_2\).
  - Take the top \(k\) distinct objects as the password.

- Why does it work?
  - No theoretical explanation, but Yan et al.’s experiments revealed pass-object pairs tend to produce larger \(F_1-F_2\).

<table>
<thead>
<tr>
<th>Object Pairs</th>
<th>0-response</th>
<th>1-response</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1, 2)</td>
<td>28</td>
<td>24</td>
<td>+4</td>
</tr>
<tr>
<td>(1, 3)</td>
<td>32</td>
<td>26</td>
<td>+6</td>
</tr>
<tr>
<td>(n-1, n)</td>
<td>40</td>
<td>28</td>
<td>+12</td>
</tr>
</tbody>
</table>
How well does Yan et al.’s attack work to break Foxtail?

- Parameters of Foxtail: \((n,k,l) = (140,14,15)\)

- Results
  - Password recovered in about 711 authentication sessions using 2D frequency tables
  - 90% of pass-objects recovered in about 540 sessions
Our contributions

- Why does Yan et al.’s attack work?
  - Yan et al.’s 2D attack ⇒ $\delta D$ attacks ($\delta \geq 1$)
  - 1D attack works as well! ⇒ Yan et al.’s 2D attack is just a generalization of the 1D attack to 2D space!
  - A general theoretical analysis of $\delta D$ attacks

- A theoretical approach for estimating the security lower bound against $\delta D$ attacks
  - This presentation will not cover this part due to time limit.

- Two new principles of designing new protocols

- Fixes to make counting based protocols more secure against $\delta D$ attacks (so to make counting still work)
Why does Yan et al.’s attack work?

- Three equalities about each object’s occurrence frequency must hold to disable each $\delta D$ attack
  - $\xi_{pass}(0) = \xi_{decoy}(0)$
  - $\xi_{pass}(1) = \xi_{decoy}(1)$
  - $\xi_{pass}(0) - \xi_{pass}(1) = \xi_{decoy}(0) - \xi_{decoy}(1)$
- $3\delta_{max}$ equalities, but only 3 parameters $(n,k,l)$
- Yan et al.’s attack works because none of the above equalities holds when $\delta = 2!$
- $\Rightarrow$ Both theoretical and experimental analysis revealed that Foxtail can never be made absolutely secure against $\delta D$ attacks!
- 1D attack also works!
  - For the default parameter \((n,k,l)=(140,14,15)\), the password was recovered after about 7,000 authentication sessions were observed.
  - Less efficient than 2D attack, but still a theoretical threat!
- Further analysis shows when \(\delta>2\), the attack still works but the number of required sessions increases drastically.
Beyond response dependent attacks and Foxtail

- The $\delta D$ attacks discussed so far treat challenges corresponding to different response values separately.
- We can also treat all challenges equally without considering the response values.

$\Rightarrow$ Two classes of statistical attacks
- $\delta D$ RDFA = Response dependent frequency analysis
- $\delta D$ RIFA = Response independent frequency analysis

- Foxtail was designed with only 1D RIFA in mind.
- Both attacks can be applied to many other protocols (not only counting based).
Two new principles for designing protocols based on counting

1. Each *object* should be sampled *independently* with the same probability regardless of its type (pass- or decoy objects).
   - This is to prevent RIFA.

2. The *response* should be *independent* of the number of pass-objects in each challenge.
   - This is to prevent RDFA.
   - It seems contradictory, but we will see how it may not be so.
A general fix to any counting based protocols with binary responses

- Generate challenges without distinguishing between pass- and decoy objects
  - Rand-Rule: select \( l \) objects at random
  - Each object appears with the same probability \( p \) (\( l \) will be session varying if \( p < 1 \))
- Flip the response by a hidden bit (challenge)
  - The (binary) response is flipped according to a random hidden bit (which can be seen as a hidden challenge).
  - This makes responses independent of the number of pass-objects present in the challenge.
- If the response is not binary, the random hidden bit will be replaced by a random hidden variable.
How to generate the random hidden bit?

- Ideally, an out-of-band (OOB) channel can be used.
  - This idea was proposed by some other researchers at CHI 2008 to design a solution based on hidden challenges.
- If an OOB channel is not acceptable or impossible, the flip bit has to be hidden in the public challenge.
  - Below is an example for Foxtail.

First challenge

Second (or flip-bit) challenge
A fix to the fix

- The implementation of the fix without an OOB channel is actually still insecure.
  - The adversary can guess the position of the flip bit.
  - If the guess is wrong, nothing happens.
  - If the guess is correct, it will contribute to the frequency difference between pass- and decoy objects.
  - Experimentally validated, so it is a real threat.

- A possible fix to the fix
  - Use $m>1$ flip bits instead of just one.
  - When $m=k$, the adversary will have to guess the whole password so have no advantage by guessing the $m$ bits.
  - Usability suffers: authentication time increases.
Yet another (less generic) fix to Foxtail protocol (1)

- Foxtail 2.1: The fixed Foxtail protocol
  - All objects appear in each challenge.
  - Each object is assigned a random weight in \{0,1,2,3\}.
  - The response function is changed to the sum of the weights of all pass-objects mod 4.

- Is this enhanced Foxtail secure?
  - Secure against \(\delta\)D RIFA for any \(1 \leq \delta \leq k\).
  - Secure against \(\delta\)D RDFA when \(\delta < k\).
  - “Insecure” against \(k\)D RDFA, but in this case the attacking complexity is the same as brute forcing the password. \(\Rightarrow\) Secure against \(k\)D RDFA as well.

- Usability suffers: challenges are large.
Yet another (less generic) fix to Foxtail protocol (2)

- Foxtail 2.2: The fixed Foxtail protocol
  - Only $l$ objects appear in each challenge.
  - Each object is assigned a random weight in $\{0,1,2,3\}$.
  - Rand-Rule is used to select the $l$ objects.
  - The response function is changed to the sum of the weights of all pass-objects mod 4.

- Is this enhanced Foxtail secure?
  - Secure against $\delta$D RIFA for any $1 \leq \delta \leq k$.
  - Theoretically insecure against $\delta$D RDFA for any $1 \leq \delta \leq k$.
  - $>2,000$ sessions are needed to launch a successful attack when $(n,k,l)=(140,14,20)$. $\Rightarrow$ Practically secure!

- Usability improves: challenges are smaller.
- At NDSS 2012 Yan et al. also proposed a framework for estimating usability of human authentication protocols without running any real user study.

- The estimated authentication times
  - Original insecure Foxtail: 213 seconds
  - Foxtail 2.1: 475 seconds
  - Foxtail 2.2: 274 seconds

- Foxtail 2.2 is practical secure and slightly less usable than the original Foxtail.

- Open questions for future work: 1) are there other attacks to Foxtail 2.x? 2) how can we do better?
Thanks for your attention!