Backdoored Implementation of Stern's Zero-Knowledge Identification Protocol

Gaëtan Leurent

Inria, France
Gaetan.Leurent@inria.fr

1 Introduction

Stern described a code-based zero-knowledge identification scheme in 1993 [6], which became the basis of several improved variants [8,2,5,4]. It is quite attractive because it is provably secure, but only uses simple operations (matrix multiplications and bit permutations).

In this work, we add a backdoor to a proof-of-concept implementation from Cayrel et al.[3], with a subtle implementation flaw. The new version still accepts all legitimate provers, and reject almost all illegitimate ones. However, an adversary knowing that the flaw is present can fool the authentication. A similar backdoor can be planted in virtually any implementation of the scheme, and in most later variants.

1.1 Description of Stern's Identification Scheme

We briefly describe Stern's zero-knowledge code-based identification scheme [6]. A random $n \times k$ matrix H is given as a parameter; then each user selects a random secret key s of weight w (a fixed parameter) and length n, and publishes the identificator $i = H(s)$. The scheme also uses commitments, computed with a hash function $\mathcal{H}$.

The security of the scheme is based on the syndrome decoding problem: the parameters should be chosen so that it is infeasible to find a vector x of weight w so that $H(x) = i$. The authentication protocol is a as follows:

- The prover picks a random n-bit word y and a random permutation[1] σ of $[1, n]$. Then he sends to the verifier:

$$c_1 = \mathcal{H}(\sigma \parallel H(y)) \qquad c_2 = \mathcal{H}(\sigma(y)) \qquad c_3 = \mathcal{H}(\sigma(s \oplus y))$$

- The verifier picks a random number b in $\{0, 1, 2\}$.
- If $b = 0$:
  - The prover reveals y and σ.
  - The verifier checks c_1 and c_2.
- If $b = 1$:
  - The prover reveals $y \oplus s$ and σ.
  - The verifier checks c_1 and c_3 (note that $H(y) = H(y \oplus s) \oplus i$).
- If $b = 2$:
  - The prover reveals $\sigma(s)$ and $\sigma(y)$.
  - The verifier checks c_2 and c_3, and verifies that $\sigma(s)$ has weight w (note that $\sigma(s \oplus y) = \sigma(s) \oplus \sigma(y)$).

An honest prover is always successfully, while an adversary only succeeds with probability $2/3$ unless he knows a weight w preimage of i. The protocol is repeated until the cheating probability is negligible.

[1] A commitment $\mathcal{H}(\sigma)$ of σ can be computed using $\mathcal{H}(\sigma(1) \parallel \ldots \parallel \sigma(n))$
For an n-bit word x, we also use $\sigma(x)$ to denote the value of x with the bits permuted according to σ.

2 Our Backdoor

We use with the implementation of Stern's scheme by Cayrel et al. [3], available at `https://docs.google.com/file/d/0B4Cy03-L745ZdW8xQ2JTaWRMYzA`, and we modify the Hamming weight routine in order to introduce our backdoor. We use an implementation inspired by the HACKMEM 169 algorithm [1,7], extended to 64 bits integers. Our new function is shown in Figure 1.

```
inline static unsigned long cb(unsigned long x) {     /* count the number of  */
                                                      /* bits contained in x  */

  // 64-bit version of HAKMEM 169
  // Adapted from https://chessprogramming.wikispaces.com/Population+Count#toc20

  // Special case because 6-bit additions overflow if all bits are 1
  if (x == (unsigned long)-1)
    return 64;

   x = x  - ((x >> 1)  & 0133333333333333333333ULL)
          - ((x >> 2)  & 0111111111111111111111ULL);
   x = (x +  (x >> 3)) & 0070707070707070707070ULL ;
   return x % 63; /* casting out 63 */
}
```

Fig. 1. Flawed Hamming weight computation.
Note that the initial code assumed 64-bit `long int`, and we use the same assumption here. A better code would use `uint64_t` instead of `long int`.

2.1 The bug

The Hamming weight function seem to be an improvement: it is more efficient than the initial one and is an adaptation of a legitimate algorithm. However, it incorrectly returns 0 for inputs of weight exactly 63, because the casting out with a modulo operation overflows. This is very unlikely to be detected with random tests, or during the normal operation of the program. Still, this strongly undermines the security of the protocol.

2.2 Exploit

An adversary can exploit this bug by creating a preimage s' of the target syndrome i of weight $w + 63k$ for any k. This is much easier[2] that the initial syndrome decoding problem where the preimage should have weigh only w. Then, the adversary selects a permutation σ so that the extra bits are divided in k groups of 63, each group in a single 64-bit word. He can now win the authentication game using this value s' instead of s, and a permutation σ with the previous property. The cheating should be detected when the verifier picks $b = 2$, but since the Hamming weight computation is flawed, the verifier believes that the Hamming weight of $\sigma(s')$ is w.

[2] For instance with $n = 1024$, $k = 512$, $w = 110$, if we assume that a random preimage of i follows a binomial distribution, a preimage of weight $110 + 6 \times 63 = 488$ is found after approximately 2^7 trials.

References

1. Beeler, M., Gosper, R.W., , Schroeppel, R.: "HAKMEM", Memo 239. Tech. rep., Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Mass. (1972)
2. Cayrel, P., Gaborit, P., Galindo, D., Girault, M.: Improved identity-based identification using correcting codes. CoRR abs/0903.0069 (2009), http://arxiv.org/abs/0903.0069
3. Cayrel, P.L., Günther, F., Rother, H., Hoffman, G.: Implementation of code-based zero-knowledge identification schemes (2011), http://www.cayrel.net/?Implementation-of-code-based-zero
4. Cayrel, P.L., Véron, P., Alaoui, S.M.E.Y.: A Zero-Knowledge Identification Scheme Based on the q-ary Syndrome Decoding Problem. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 6544, pp. 171–186. Springer (2010)
5. Gaborit, P., Girault, M.: Lightweight code-based identification and signature. In: Information Theory, 2007. ISIT 2007. IEEE International Symposium on. pp. 191–195. IEEE (2007)
6. Stern, J.: A New Identification Scheme Based on Syndrome Decoding. In: Stinson, D.R. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 773, pp. 13–21. Springer (1993)
7. The CPW Team: Population count, https://chessprogramming.wikispaces.com/Population+Count#toc20
8. Véron, P.: Improved identification schemes based on error-correcting codes. Applicable Algebra in Engineering, Communication and Computing 8(1), 57–69 (1997)