Practical Key Recovery for Discrete-Logarithm Based Authentication Schemes from Random Nonce Bits

Damien Vergnaud

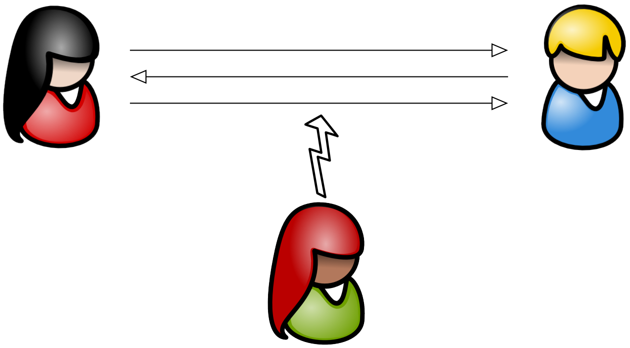
École normale supérieure

CHES
September, 15th 2015

(with Aurélie Bauer)

Contents

Identification Schemes

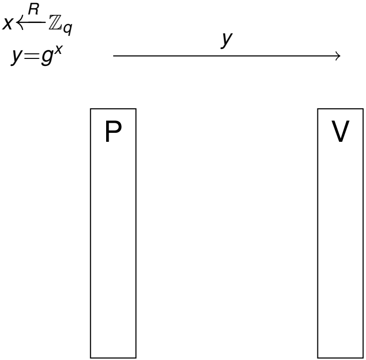


- enables a prover to identify itself to a verifier
- Adversary goal: impersonation
  - playing the role of Alice but denied the secret key,
  - it should have negligible probability of making Bob accept.
  - **passive attacks** / **active attacks**

Schnorr's Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group of prime order q

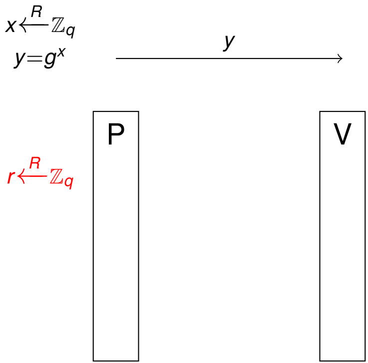
Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.



Schnorr's Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group of prime order q

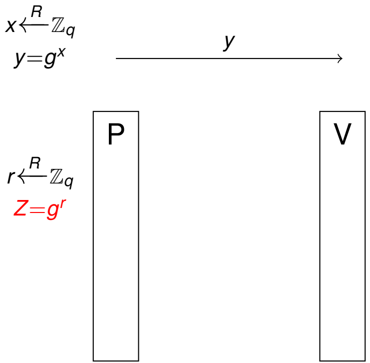
Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.



Schnorr's Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group of prime order q

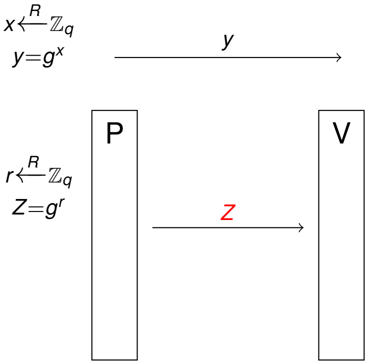
Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.



Schnorr's Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group of prime order q

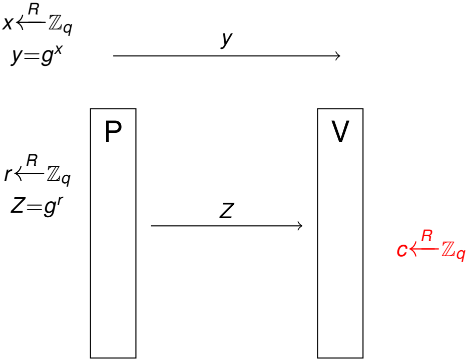
Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.



Schnorr's Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group of prime order q

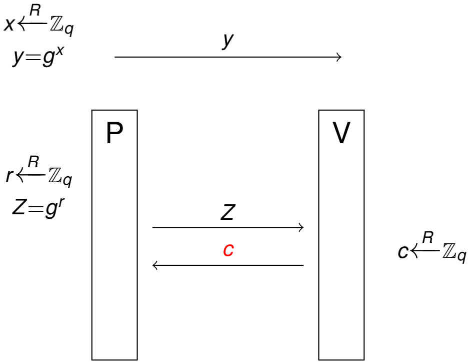
Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.



Schnorr's Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group of prime order q

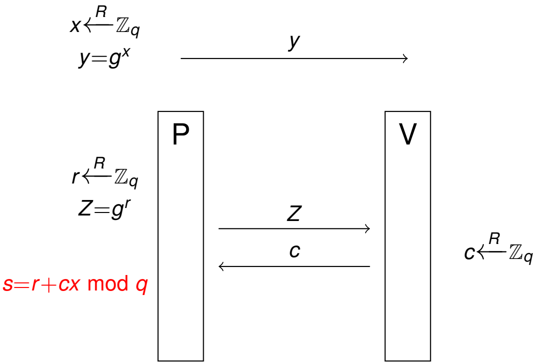
Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.



Schnorr's Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group of prime order q

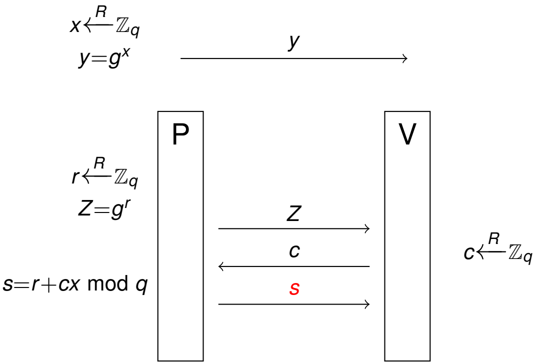
Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.



Schnorr's Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group of prime order q

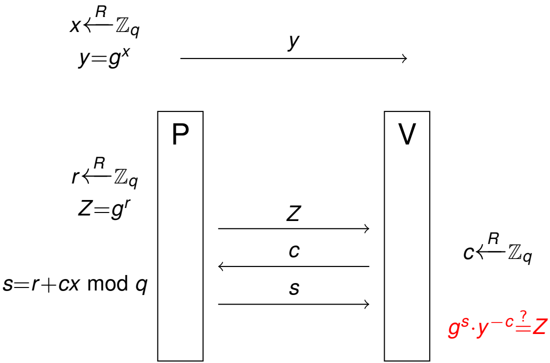
Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.



Schnorr's Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group of prime order q

Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.



GPS Identification Scheme

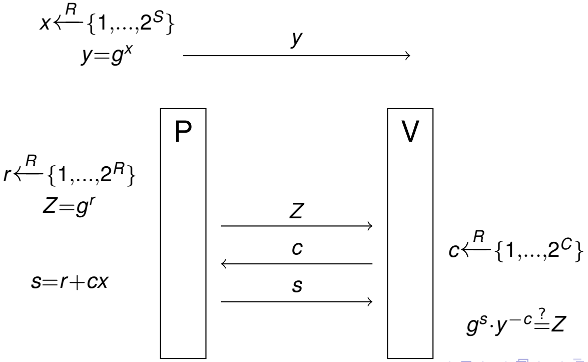
- proposed by Girault in 1991
- formally analyzed by Poupard, and Stern in 1998
- based on Schnorr's identification scheme
- Leaves modular reduction in response-calculation step
  - save computation time
  - allows fast on-the-fly authentication (use of **coupons**)
- $\rightsquigarrow$ signatures using Fiat-Shamir transform

GPS Identification Scheme

$\mathbb{G} = \langle g \rangle$ a group

Prover P proves to verifier V that it knows the discrete log x of a public group element $y = g^x$.

Parameters (128-bit security level): $(S, R, C) = (256, 512, 128)$



Cryptanalysis of DL-based Schemes

- Discrete logarithm computation of $x = \log_g(y) \rightsquigarrow$ **impersonation**
- Knowledge of $r = \log_g(Z)$
  $\rightsquigarrow$ Key recovery: $s = r + cx \Rightarrow x = (s - r)/c \rightsquigarrow$ **impersonation**
- This knowledge may be due to
  - a weak random number generator
  - a timing attack
  - a probing attack
  - ...

Cryptanalysis of DL-based Schemes

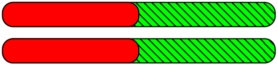
- Discrete logarithm computation of $x = \log_g(y) \rightsquigarrow$ **impersonation**
- Knowledge of $r = \log_g(Z)$
  $\rightsquigarrow$ Key recovery: $s = r + cx \Rightarrow x = (s - r)/c \rightsquigarrow$ **impersonation**
- This knowledge may be due to
  - a weak random number generator
  - a timing attack
  - a probing attack
  - ...

Cryptanalysis of DL-based Schemes

- Discrete logarithm computation of $x = \log_g(y) \rightsquigarrow$ **impersonation**
- Knowledge of $r = \log_g(Z)$
  $\rightsquigarrow$ Key recovery: $s = r + cx \Rightarrow x = (s - r)/c \rightsquigarrow$ **impersonation**
- This knowledge may be due to
  - a weak random number generator
  - a timing attack
  - a probing attack
  - …

Cryptanalysis of DL-based Schemes

- Kuwakado, Tanaka (1999):
  half of *r*'s LSB leaked for two identification/signatures

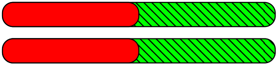


- Howgrave-Graham, Smart, Nguyen, Shparlinski (2001-2002):
  fraction of *r*'s consecutive bits for several identification/signatures

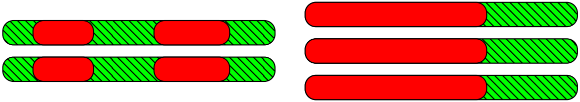
- Our work:
  fraction of *r*'s bits for several identification/signatures
  not necessarily consecutive

Cryptanalysis of DL-based Schemes

- Kuwakado, Tanaka (1999):
  half of r's LSB leaked for two identification/signatures



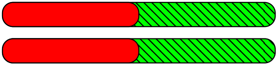
- Howgrave-Graham, Smart, Nguyen, Shparlinski (2001-2002):
  fraction of r's consecutive bits for several identification/signatures



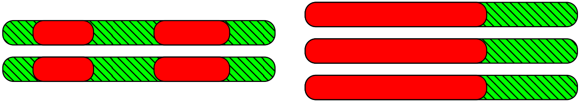
- Our work:
  fraction of r's bits for several identification/signatures
  not necessarily consecutive

Cryptanalysis of DL-based Schemes

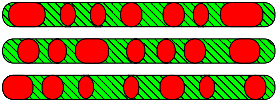
- Kuwakado, Tanaka (1999):
  half of r's LSB leaked for two identification/signatures



- Howgrave-Graham, Smart, Nguyen, Shparlinski (2001-2002):
  fraction of r's consecutive bits for several identification/signatures



- Our work:
  fraction of r's bits for several identification/signatures
  not necessarily consecutive



Our Work

- reconstructing private keys given a random fraction of nonce bits
  - elementary and does not make use of the lattice techniques
  - similar to reconstruction of RSA secret key
    (Heninger *et al.* Crypto'09 + Crypto'10)
- specialized to the case where the value $r + cx$ is known over $\mathbb{Z}$
  - GPS identification under **passive attacks**
  - GPS signature (Fiat-Shamir heuristic)
  - Schnorr identification under **active attacks** (small challenge)
- analysis of the algorithm's runtime behavior
- algorithm implemented (extensive experiments using it)

General Idea – Two Signatures

$$\begin{aligned} r_1 + c_1 x &= s_1 \\ r_2 + c_2 x &= s_2 \end{aligned}$$

- GOAL: reconstruct bits of nonces starting at the LSB.
- APPROACH (odd c_1 and c_2)
  - 4 choices for each pair of bits $(r_1[i], r_2[i]) \rightsquigarrow \#$ Search space: 2^{2R}
  - reduces to 2 as the relation

    $$c_2 r_1 - c_1 r_2 = c_2 s_1 - c_1 s_2 = C$$

    gives

    $$r_1[i] + r_2[i] = (C - c_2 r_1[0..i-1] - c_1 r_2[0..i-1])[i] \bmod 2$$

    $\rightsquigarrow \#$ Search space: 2^R (same as exhaustive search!)
- IDEA: Search tree can be pruned if we know some bits of r_1, r_2

General Idea – Two Signatures

$$r_1 + c_1 x = s_1$$
$$r_2 + c_2 x = s_2$$

- GOAL: reconstruct bits of nonces starting at the LSB.
- APPROACH (odd c_1 and c_2)
  - 4 choices for each pair of bits $(r_1[i], r_2[i]) \rightsquigarrow \#$ Search space: 2^{2R}
  - reduces to 2 as the relation

$$c_2 r_1 - c_1 r_2 = c_2 s_1 - c_1 s_2 = C$$

gives

$$r_1[i] + r_2[i] = (C - c_2 r_1[0..i-1] - c_1 r_2[0..i-1])[i] \bmod 2$$

$\rightsquigarrow \#$ Search space: 2^R (same as exhaustive search!)

- IDEA: Search tree can be pruned if we know some bits of r_1, r_2

General Idea – Two Signatures

$$r_1 + c_1 x = s_1$$
$$r_2 + c_2 x = s_2$$

- GOAL: reconstruct bits of nonces starting at the LSB.
- APPROACH (odd c_1 and c_2)
  - 4 choices for each pair of bits $(r_1[i], r_2[i]) \rightsquigarrow \#$ Search space: 2^{2R}
  - reduces to 2 as the relation

    $$c_2 r_1 - c_1 r_2 = c_2 s_1 - c_1 s_2 = C$$

    gives

    $$r_1[i] + r_2[i] = (C - c_2 r_1[0..i-1] - c_1 r_2[0..i-1])[i] \bmod 2$$

    $\rightsquigarrow \#$ Search space: 2^R (same as exhaustive search!)
- IDEA: Search tree can be pruned if we know some bits of r_1, r_2

Solution Tree: Example

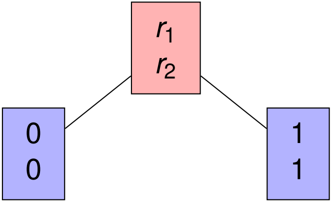
$c_1 = 9$, $s_1 = 147$

$c_2 = 15$, $s_2 = 239$

$C = 54$

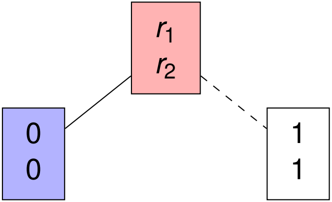
$r_1 = 1???$, $r_1 = ??10$

$$\boxed{c_2 r_1 - c_1 r_2 = C}$$



Solution Tree: Example

$$c_1 = 9,\ s_1 = 147$$
$$c_2 = 15,\ s_2 = 239$$
$$C = 54$$
$$r_1 = 1???,\ r_1 = ??10$$
$$\boxed{c_2 r_1 - c_1 r_2 = C}$$



Solution Tree: Example

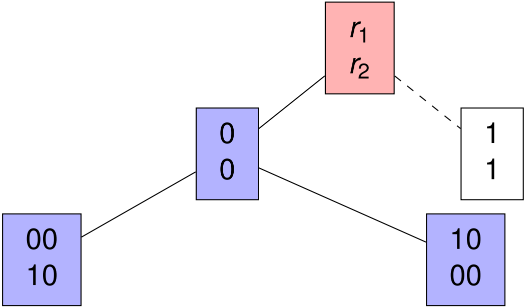
$c_1 = 9,\ s_1 = 147$

$c_2 = 15,\ s_2 = 239$

$C = 54$

$r_1 = 1???,\ r_1 = ??10$

$$\boxed{c_2 r_1 - c_1 r_2 = C}$$



Solution Tree: Example

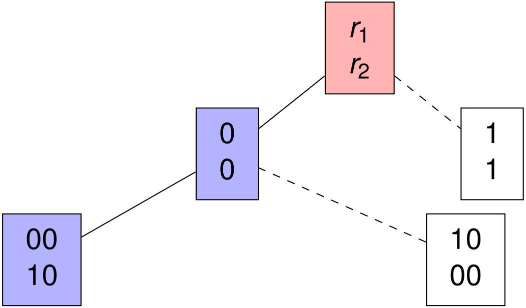
$c_1 = 9,\ s_1 = 147$

$c_2 = 15,\ s_2 = 239$

$C = 54$

$r_1 = 1???,\ r_1 = ??10$

$$\boxed{c_2 r_1 - c_1 r_2 = C}$$



Solution Tree: Example

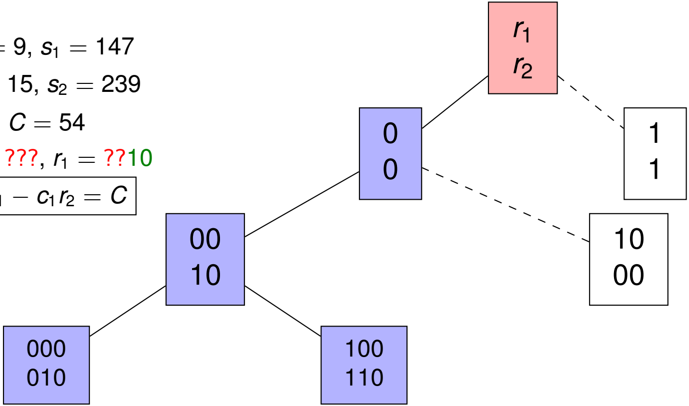
$c_1 = 9,\ s_1 = 147$

$c_2 = 15,\ s_2 = 239$

$C = 54$

$r_1 = 1???,\ r_1 = ??10$

$$\boxed{c_2 r_1 - c_1 r_2 = C}$$



Solution Tree: Example

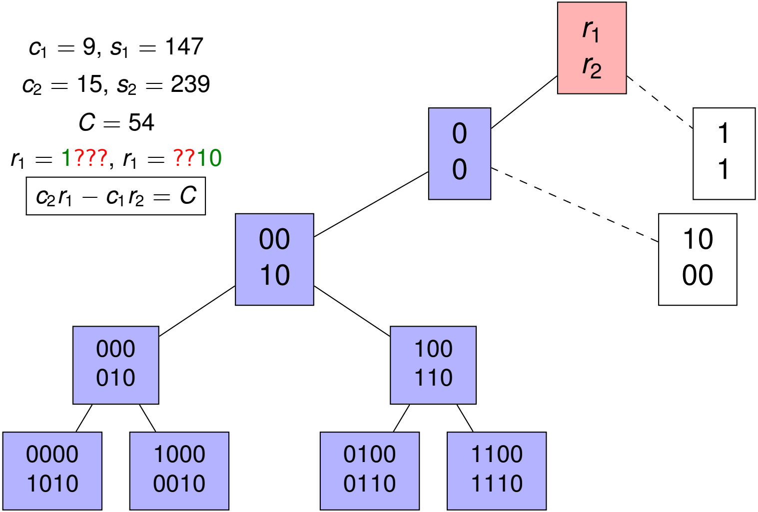
$c_1 = 9,\ s_1 = 147$

$c_2 = 15,\ s_2 = 239$

$C = 54$

$r_1 = 1???,\ r_1 = ??10$

$$\boxed{c_2 r_1 - c_1 r_2 = C}$$



Solution Tree: Example

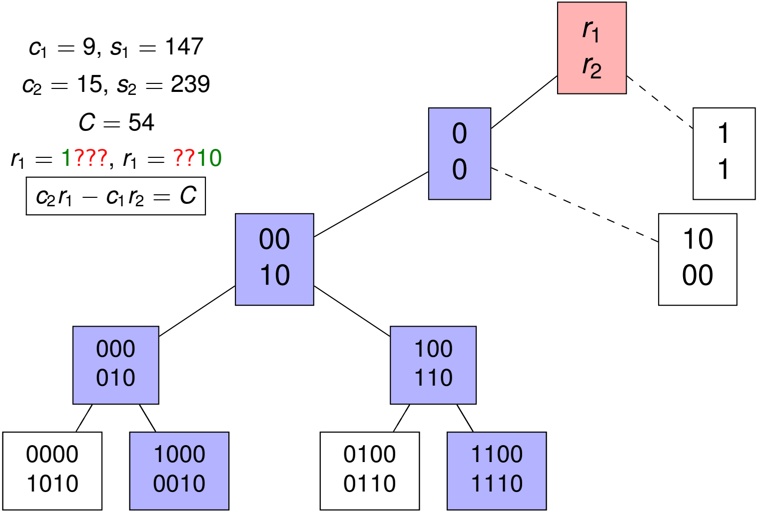
$c_1 = 9,\ s_1 = 147$

$c_2 = 15,\ s_2 = 239$

$C = 54$

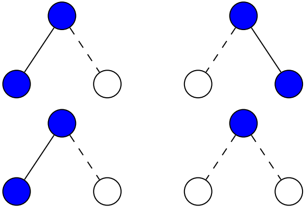
$r_1 = 1???,\ r_1 = ??10$

$$c_2 r_1 - c_1 r_2 = C$$



Branching Analysis – Two Signatures

- $r_1[i]$ or $r_2[i]$ is known
  $\rightsquigarrow$ the equation fixes the other bit.
- $r_1[i]$ and $r_2[i]$ known
  $\rightsquigarrow$ the equation is either satisfied or not.



Assumption: δ-fraction of r_1 and r_2 bits known

- $\#\{r_1[i], r_2[i] \text{ known}\} = 0$: 2 Branches, Prob = $(1-\delta)^2$
- $\#\{r_1[i], r_2[i] \text{ known}\} = 1$: 1 Branch , Prob = $2\delta(1-\delta)$
- $\#\{r_1[i], r_2[i] \text{ known}\} = 2$: γ Branch , Prob = δ^2 for $0 < \gamma < 1$

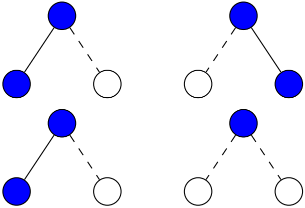
Expected number of branches from each node:

$$2 \cdot (1-\delta^2) + 1 \cdot 2\delta(1-\delta) + \gamma \cdot \delta^2 = 2 - 2\delta + \gamma\delta^2$$

Branching Analysis – Two Signatures

- $r_1[i]$ or $r_2[i]$ is known
  $\rightsquigarrow$ the equation fixes the other bit.

- $r_1[i]$ and $r_2[i]$ known
  $\rightsquigarrow$ the equation is either satisfied or not.



Assumption: δ-fraction of r_1 and r_2 bits known

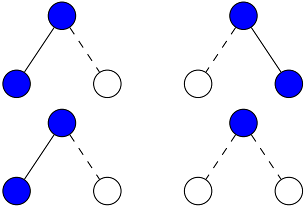
- $\#\{r_1[i], r_2[i] \text{ known}\} = 0$: 2 Branches, Prob = $(1-\delta)^2$
- $\#\{r_1[i], r_2[i] \text{ known}\} = 1$: 1 Branch , Prob = $2\delta(1-\delta)$
- $\#\{r_1[i], r_2[i] \text{ known}\} = 2$: γ Branch , Prob = δ^2 for $0 < \gamma < 1$

Expected number of branches from each node:

$$2 \cdot (1-\delta^2) + 1 \cdot 2\delta(1-\delta) + \gamma \cdot \delta^2 = 2 - 2\delta + \gamma\delta^2$$

Branching Analysis – Two Signatures

- $r_1[i]$ or $r_2[i]$ is known
  $\rightsquigarrow$ the equation fixes the other bit.
- $r_1[i]$ and $r_2[i]$ known
  $\rightsquigarrow$ the equation is either satisfied or not.



Assumption: δ-fraction of r_1 and r_2 bits known

- $\#\{r_1[i], r_2[i] \text{ known}\} = 0$: 2 Branches, Prob = $(1-\delta)^2$
- $\#\{r_1[i], r_2[i] \text{ known}\} = 1$: 1 Branch , Prob = $2\delta(1-\delta)$
- $\#\{r_1[i], r_2[i] \text{ known}\} = 2$: γ Branch , Prob = δ^2 for $0 < \gamma < 1$

Expected number of branches from each node:

$$2 \cdot (1-\delta^2) + 1 \cdot 2\delta(1-\delta) + \gamma \cdot \delta^2 = 2 - 2\delta + \gamma\delta^2$$

Branching Analysis (simplified) – Two Signatures

Growth factor of the Search Tree: $2 - 2\delta + \gamma\delta^2$

- Polynomial time attack ?
  $\rightsquigarrow$ Keep the growth factor $\simeq 1$ to restrict growth.

$$\delta = (1 - \sqrt{1-\gamma})/\gamma$$

- Experimental observation: $\gamma \simeq 1/2$ (open problem)

$$\delta \simeq 2 - \sqrt{2} \simeq 0,5857$$

For $\delta > 2 - \sqrt{2}$, the algorithm recovers the secret key in expected quadratic time.
(assuming that the effect of a bit error during reconstruction is propagated uniformly through subsequent bits of the key

Branching Analysis (simplified) – Two Signatures

Growth factor of the Search Tree: $2 - 2\delta + \gamma\delta^2$

- Polynomial time attack ?
  $\rightsquigarrow$ Keep the growth factor $\simeq 1$ to restrict growth.

$$\delta = (1 - \sqrt{1-\gamma})/\gamma$$

- Experimental observation: $\gamma \simeq 1/2$ (open problem)

$$\delta \simeq 2 - \sqrt{2} \simeq 0,5857$$

For $\delta > 2 - \sqrt{2}$, the algorithm recovers the secret key in expected quadratic time.
(assuming that the effect of a bit error during reconstruction is propagated uniformly through subsequent bits of the key

Branching Analysis (simplified) – n Signatures

Assumption: δ-fraction of $r_1, \ldots, r_n$ bits known

- $\#\{r_1[i], \ldots, r_n[i] \text{ known}\} = 0$: 2 Branches, Prob = $(1-\delta)^n$
- $\#\{r_1[i], \ldots, r_n[i] \text{ known}\} = 1$: 1 Branches, Prob = $n\delta(1-\delta)^{n-1}$
- $\#\{r_1[i], \ldots, r_n[i] \text{ known}\} = 2$: γ_1 Branches, Prob = $\binom{n}{2}\delta^2(1-\delta)^{n-2}$
- ...
- $\#\{r_1[i], \ldots, r_n[i] \text{ known}\} = n$: γ_{n-1}, Prob = δ^n
- Experimental observation: $\gamma_i \simeq 2^{-i}$ (open problem)

For $\delta > 2 - 2^{1-1/n} \simeq \ln(2)/n$, the algorithm recovers the secret key in $O(nk^2)$ expected time.
(assuming that the effect of a bit error during reconstruction is propagated uniformly through subsequent bits of the key

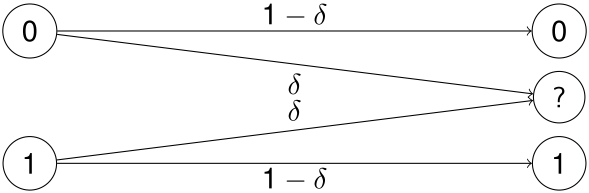
Branching Analysis (simplified) – n Signatures

Assumption: δ-fraction of $r_1, \ldots, r_n$ bits known

- $\#\{r_1[i], \ldots, r_n[i] \text{ known}\} = 0$: 2 Branches, Prob = $(1-\delta)^n$
- $\#\{r_1[i], \ldots, r_n[i] \text{ known}\} = 1$: 1 Branches, Prob = $n\delta(1-\delta)^{n-1}$
- $\#\{r_1[i], \ldots, r_n[i] \text{ known}\} = 2$: γ_1 Branches, Prob = $\binom{n}{2}\delta^2(1-\delta)^{n-2}$
- ...
- $\#\{r_1[i], \ldots, r_n[i] \text{ known}\} = n$: γ_{n-1}, Prob = δ^n
- Experimental observation: $\gamma_i \simeq 2^{-i}$ (open problem)

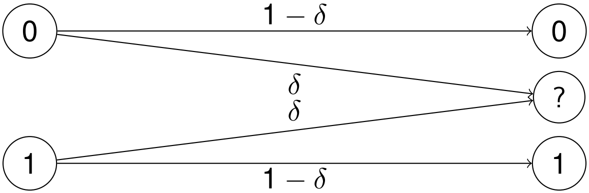
For $\delta > 2 - 2^{1-1/n} \simeq \ln(2)/n$, the algorithm recovers the secret key in $O(nk^2)$ expected time.
(assuming that the effect of a bit error during reconstruction is propagated uniformly through subsequent bits of the key

Binary Erasure Channel



- Channel capacity: $1 - \delta$
- **Code C:** set of 2^r words on nr bits (r Hensel lifts w/o any pruning) $\rightsquigarrow$ **Code rate:** $1/n$
- **Received word:** noisy version of the nonces.

Binary Erasure Channel



- Channel capacity: $1 - \delta$
- **Code C:** set of 2^r words on nr bits (r Hensel lifts w/o any pruning)
  $\rightsquigarrow$ **Code rate:** $1/n$
- **Received word:** noisy version of the nonces.

Shannon's noisy-channel coding theorem

Reliable decoding impossible when the code rate exceeds the capacity.

$\rightsquigarrow$ Variants of the algorithm cannot be efficient for $\delta \leq 1/n$

What about errors instead of erasures?

- **Scenario:** Attacker gets all bits but errors occur
  - i.e. we obtain erroneous versions of nonces
- **Motivation:** Physical measurements induces random faults.

The adversary knows $r'_1, \ldots, r'_n$ s.t.

$$\Pr(r'_j[i] = r_j[i]) = 1 - \delta, \text{ for all } i, j$$

(for simplicity, we assume δ is known)

Information provided by the Oracle is no longer fault-free!

Can we adapt the previous algorithm?

- The previous pruning algorithm requires correct bits.
  - otherwise we might prune the correct solution
- Need pruning with the following properties:
  - Correct key survives with large probability.
  - Sufficiently many incorrect keys are pruned.
  - similar to Henecka-May-Meurer error correction in RSA secret keys (Crypto'10)
- IDEA: **Use many subsequent bits instead of just one**
  - grow subtrees of depth t
  - prune leaves whose Hamming distance is greater than some threshold d

Can we adapt the previous algorithm?

- The previous pruning algorithm requires correct bits.
  - otherwise we might prune the correct solution
- Need pruning with the following properties:
  - Correct key survives with large probability.
  - Sufficiently many incorrect keys are pruned.
  - similar to Henecka-May-Meurer error correction in RSA secret keys (Crypto'10)
- IDEA: **Use many subsequent bits instead of just one**
  - grow subtrees of depth t
  - prune leaves whose Hamming distance is greater than some threshold d

Can we adapt the previous algorithm?

- The previous pruning algorithm requires correct bits.
  - otherwise we might prune the correct solution
- Need pruning with the following properties:
  - Correct key survives with large probability.
  - Sufficiently many incorrect keys are pruned.
  - similar to Henecka-May-Meurer error correction in RSA secret keys (Crypto'10)
- IDEA: **Use many subsequent bits instead of just one**
  - grow subtrees of depth t
  - prune leaves whose Hamming distance is greater than some threshold d

Analysis of Error-Correction

- GOALS:
  - $\#$ of nodes polynomially bounded (t not too large, i.e. $t = O(\log r)$)
  - Separate correct and incorrect partial solutions (t large)
  - Correct solution passes all pruning steps (d not too large)
  - Few incorrect solutions survive pruning (d large)
- **Analysis (see paper):** for $\epsilon > 0$
  - $t = \ln(R)/n\epsilon^2$
  - $\gamma = \sqrt{(1 + 1/t)\ln(2)/2n}$
  - $d = nt(1/2 + \gamma)$
  - $\delta = 1/2 - \gamma - \epsilon$

Analysis of Error-Correction

- GOALS:
  - $\#$ of nodes polynomially bounded (t not too large, i.e. $t = O(\log r)$)
  - Separate correct and incorrect partial solutions (t large)
  - Correct solution passes all pruning steps (d not too large)
  - Few incorrect solutions survive pruning (d large)
- **Analysis (see paper):** for $\epsilon > 0$
  - $t = \ln(R)/n\epsilon^2$
  - $\gamma = \sqrt{(1 + 1/t)\ln(2)/2n}$
  - $d = nt(1/2 + \gamma)$
  - $\delta = 1/2 - \gamma - \epsilon$

Cryptanalytic Result

For $\epsilon > 0$ and $\delta > \frac{1}{2} - \sqrt{\frac{\ln(2)}{2n}} - \epsilon$, the algorithm recovers the secret key in $O(nk^{2+\ln(2)/n\epsilon^2})$ expected time.
(assuming that the effect of a bit error during reconstruction is propagated uniformly through subsequent bits of the key

n	2	3	4	5	6	n
δ	0.084	0.160	0.205	0.237	0.260	$1/2 - \sqrt{\ln(2)/2n}$
δ^*	0.110	0.174	0.214	0.243	0.264	$H_2^{-1}(1 - 1/n)$

Conclusion

- Key recovery attack on DL-based authentication schemes
  - given a random fraction of nonce bits
  - given all bits with noise
- The two approaches can be combined (and also with other side information)
- Open problems:
  - Combine these algorithms with discrete-log algorithms with partial knowledge
  - Adapt to schemes with modular reduction (using leakage of modular reduction ?)