

# KryptonIT: The Discovery of a New Cryptographic System for Storing Secrets

Reto E. Koenig

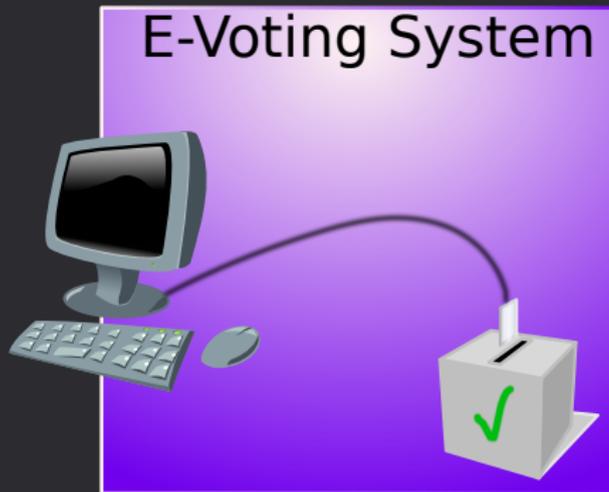
University of Fribourg  
&  
Bern University of Applied Sciences

25.03.2011



What you should know about building new cryptographic systems  
Ronald Rivest: “Never build your own cryptographic system!”

## Current E-Voting Systems



## Voter's view on a current E-Voting Systems

- It is Understandable
- It is Simple to Use
- Fast
- “Cheap”



## Adversary's view on a current E-Voting Systems

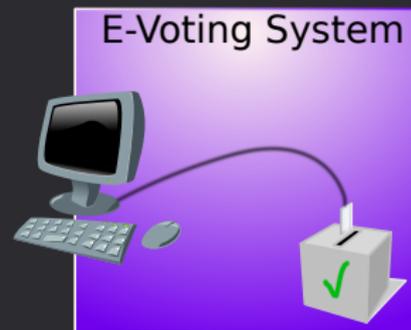
*Do not Underestimate the Power of the Coercer*

- You Shall Not Vote
  - You Shall Vote
  - You Shall Vote As I Say
  - I randomize Your Vote
  - I Vote for You
- 
- I am Watching You
  - I Know How You Voted Last Summer
- 
- It is Fast
  - It is Scalable
  - It is "Cheap"

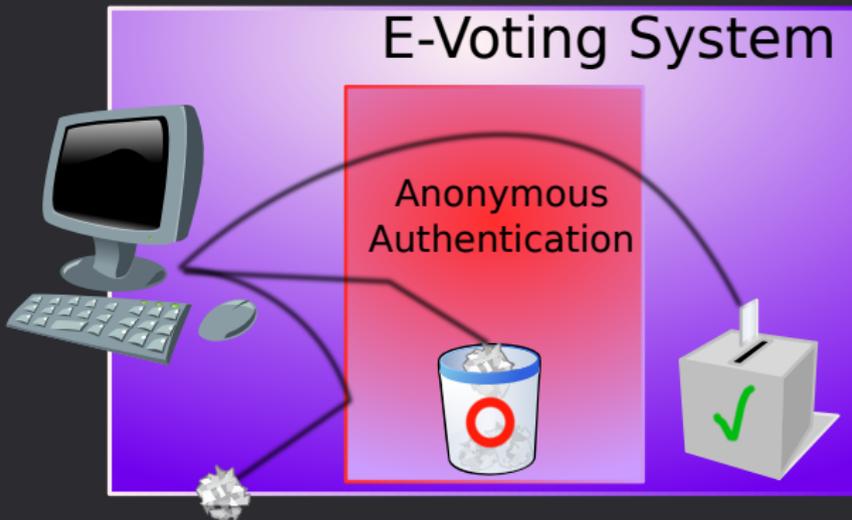


## Overall view on a current E-Voting Systems

- It is Simple
- It is Fast
- It is Cheap
- It is completely insecure



## Cryptographer's new Hope



## Cryptographer's view on their E-Voting Systems

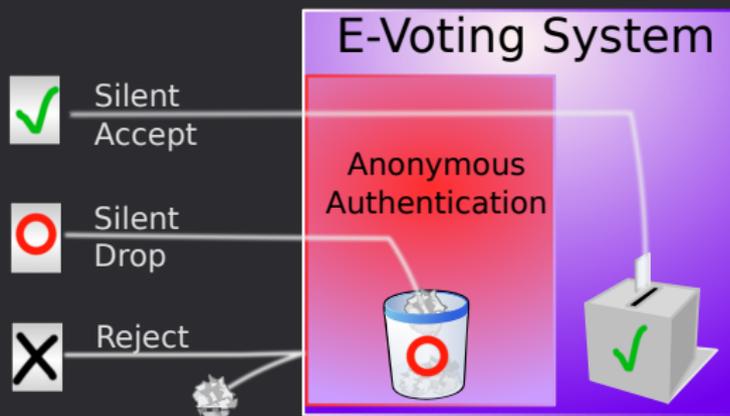
- It is Coercion-Resistant
- It is End-To-End Verifiable
- **It is secure**
- The voter has to remember "some" "secrets"



## Some Secrets?

E2E verifiable coercion resistant E-Voting systems require:

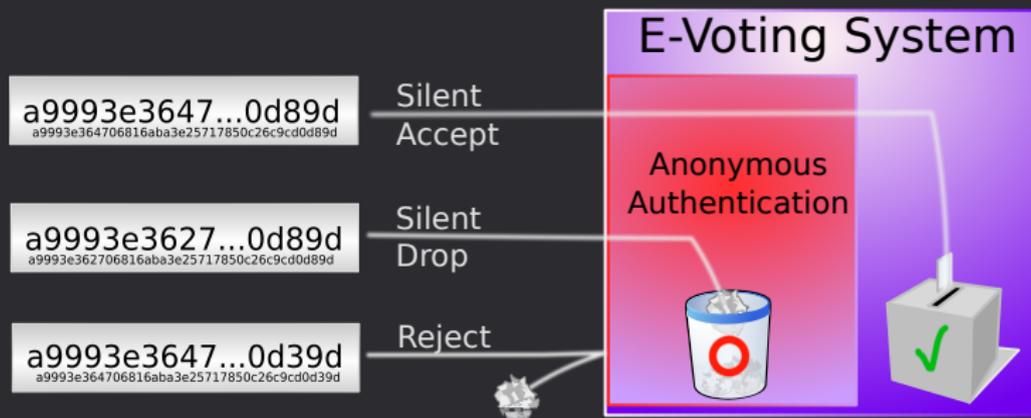
- 3 - different kind of credentials with very high entropy
- kept top secret by each voter



## Some Secrets?

E2E verifiable coercion resistant E-Voting systems require:

- 3 - different kind of credentials with very high entropy
- kept top secret by each voter



## A Realistic Example With 20 Credentials

In order to render the E-Voting System coercion resistant, each voter...

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

924f61661a3472da74307a35f2c8d22e07e84a4d	○
cbf019b764b9477080c5a9a748a2911a5fa6d614	○
fc0ccd6641d45ef2efd926c3a6f7f3ac268e9e3	✓
a29965fbb2954c8a66d856e8eb891bce5f49dacf	✓
3a710d2a84f856bc4e1c0bb93ca517893c48691	○
e1a5b5d17e51d56f0d6fcc060968ff238afb9b32	○
cbf019b764b9477080caa9a748a2911a5fa6d614	○
a29965fbb2954c8a66d8b6e8eb891bce5f49dacf	○
22eb60281c37e6611e85e7a432a45c8f3525749	○
924f61661a34d2da74307a35f2c8d22e07e84a4d	●
f5abae5832976490847c13be2c54bcfb3260f0f3	○
b1aa98ad3a02ffe896c49687300d8644f50fdd88	✓
2789a4eb84e43e4d5b60d87b3d1edec02d4c449e	○
2789a4eb84e43e4d5b60d87d3d1edec02d4c449e	○
fc0ccd6641d45ef2efd946c3a6f7f3ac268e9e3	●
e1a5b5d17e51d56f0d6fcc060968ff238afb9b32	○
fd8b823d965947fc7d9f470907ca18ed68243557	○
f5abae583297649547c13be2c54bcfb3260f0f3	✓
3a710d2a84f856bce1c0bb93ca517893c48691	✓
b1aa98ad3a02ffe896c49687300d8644f50fdd88	✓
22eb60281c37e6611e85e7a432a45c8f3525749	○
fd8b823d965947fc7d9f470907ca18ed68243557	○

<sup>a</sup> E-voting system accepts both without returning any hint

## A Realistic Example With 20 Credentials

In order to render the E-Voting System coercion resistant, each voter...

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

```

924f61661a3472da74307a35f2c8d22e07e84a4d
cbf019b764b9477080c5a9a748a2911a5fa6d614
fc0ccd6641d45ef2efd926c3a6f7f3ac268e9e3
a29965fbb2954c8a66d856e8eb891bce5f49dacf
3a718d2a84f856bc4e1c0bb93ca517893c48691
e1a5b5d17e51d56f0d6fcc060968ff238afba9b32
cbf019b764b9477080caa9a748a2911a5fa6d614
a29965fbb2954c8a66d8b6e8eb891bce5f49dacf
22eb60281c37e6611e85e7a432a45c8f3525749
924f61661a34d2da74307a35f2c8d22e07e84a4d
f5abae5832976490847c13be2c54bcfb3260f0f3
b1aa98ad3a02ffe896c49687300d8644f50fdd88
2789a4eb84e43e4d5b60d87b3d1edec02d4c449e
2789a4eb84e43e4d5b60d87d3d1edec02d4c449e
fc0ccd6641d45ef2efd946c3a6f7f3ac268e9e3
e1a5b5d17e51d56f0d6fcc060e68ff238afba9b32
fd8b823d965947fc7d9f470907ca18ed68243557
f5abae583297649547c13be2c54bcfb3260f0f3
3a718d2a84f856bce1c0bb93ca517893c48691
b1aa98ad3a02ffe896c49687300d8644f50fdd88
22eb60281c37e6611e85e7a432a45c8f3525749
fd8b823d965947fc7d9f470907ca18ed68243557

```

<sup>a</sup> E-voting system accepts both without returning any hint

## A Realistic Example With 20 Credentials

In order to render the E-Voting System coercion resistant, each voter...

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



<sup>a</sup> E-voting system accepts both without returning any hint

But How Should the Voter Store those Secrets?

Indeed it is not possible to store these secrets with some keys (passwords) in a current crypto-systems. (→ Discussion)

## Our View on the Secret-Storage System

### The system...

- ...allows to choose freely  $n$  (usually low entropy) keys
- ...allows to choose freely  $n$  (usually high entropy) secrets
- ...has to store multiple secrets in one storage (aka cipher)
- ...has to retrieve only the secret correlated to the key
- ...has to have all properties of a (symmetric) crypto-system



Let's get Formal...

### Prerequisites

$\mathcal{S}$  = *secret space*, set of all possible secrets (typically high-entropy)

$\mathcal{K}$  = *key space*, set of all possible keys (typically low-entropy)

$S = (s_1, \dots, s_n)$ ,  $s_i \in \mathcal{S}$ , an  $n$ -tuple of (not necessarily distinct) secrets ( $n \geq 1$ )

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

$\mathcal{C}$  = *storage space*, the set of all possible storages

$c$  = a particular storage

## Functions of the Secret-Storing System

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

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

$$\text{retrieve} : \mathcal{C} \times \mathcal{K} \longrightarrow \mathcal{S}$$

the *retrieval function*

---

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

$\text{retrieve}_{k_i}(c) = s_i \in \mathcal{S}$  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 traditional 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  $\frac{1}{|S|}$

## Definition

A *secret-storing system* of order  $n$ ,

$$\Sigma = (\mathcal{S}, \mathcal{K}, \mathcal{C}, \text{store}, \text{retrieve}),$$

consists of a secret space  $\mathcal{S}$ , a key space  $\mathcal{K}$ , a storage space  $\mathcal{C}$ , and two functions `store` and `retrieve` with properties as introduced above.

## An Example in $\mathbb{R}$ (Storing)

$$S = (s_1, \dots, s_i, \dots, s_n)$$

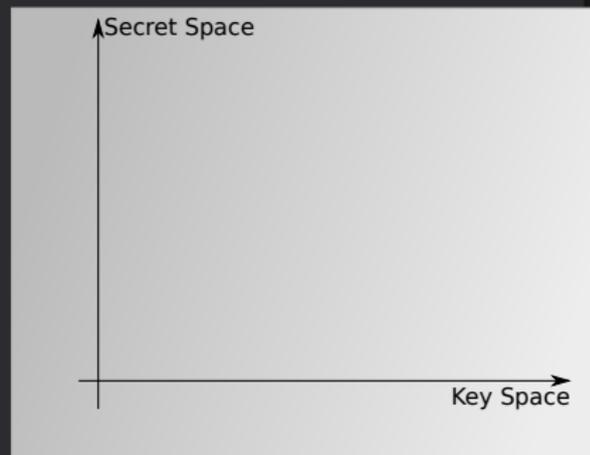
$$K = (k_1, \dots, k_i, \dots, k_n)$$

$$c = \text{store}_K(S)$$

$$c = \mathcal{A}_K^*(S)$$

$$c = f = \sum_{z=0}^t a_z x^z$$

\* Lagrange interpolation



## An Example in $\mathbb{R}$ (Storing)

$$S = (s_1, \dots, s_i, \dots, s_n)$$

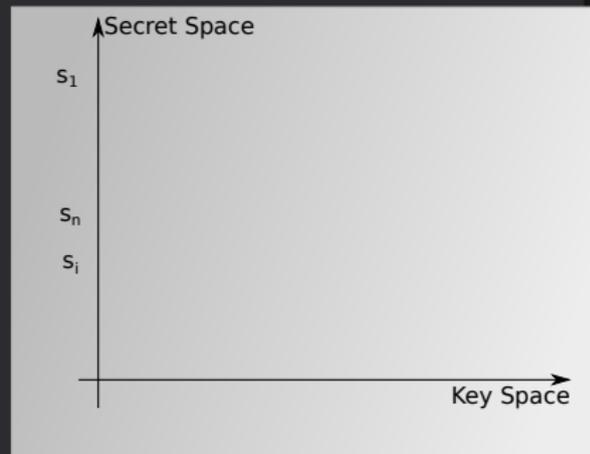
$$K = (k_1, \dots, k_i, \dots, k_n)$$

$$c = \text{store}_K(S)$$

$$c = \mathcal{A}_K^*(S)$$

$$c = f = \sum_{z=0}^t a_z x^z$$

\* Lagrange interpolation



## An Example in $\mathbb{R}$ (Storing)

$$S = (s_1, \dots, s_i, \dots, s_n)$$

$$K = (k_1, \dots, k_i, \dots, k_n)$$

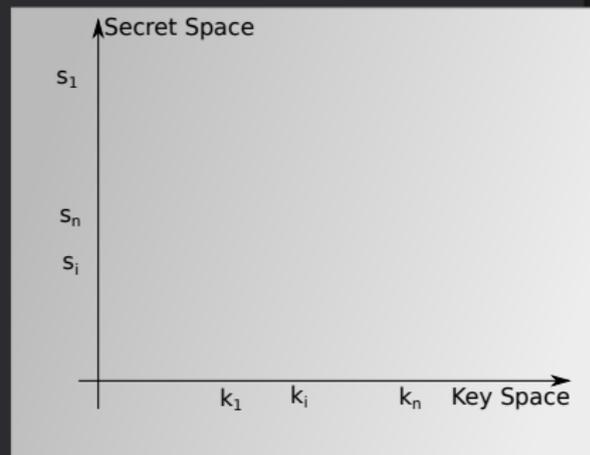
$$c = \text{store}_K(S)$$

$$c = \Lambda_K^*(S)$$

$$c = f = \sum_{z=0}^t a_z x^z$$

---

\*Lagrange interpolation



## An Example in $\mathbb{R}$ (Storing)

$$S = (s_1, \dots, s_i, \dots, s_n)$$

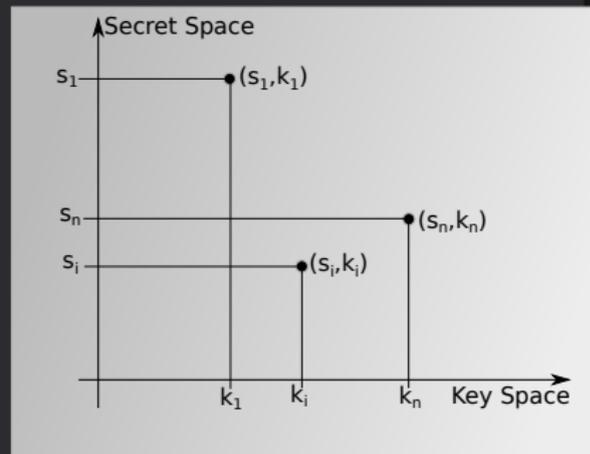
$$K = (k_1, \dots, k_i, \dots, k_n)$$

$$c = \text{store}_K(S)$$

$$c = \Lambda_K^*(S)$$

$$c = f = \sum_{z=0}^t a_z x^z$$

\*Lagrange interpolation



## An Example in $\mathbb{R}$ (Storing)

$$S = (s_1, \dots, s_i, \dots, s_n)$$

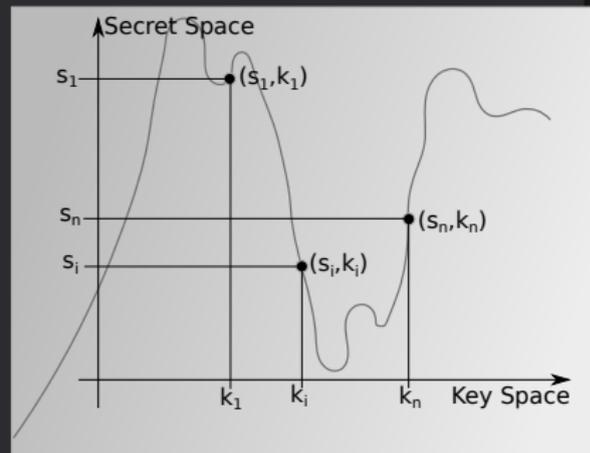
$$K = (k_1, \dots, k_i, \dots, k_n)$$

$$c = \text{store}_K(S)$$

$$c = \Lambda_K^*(S)$$

$$c = f = \sum_{z=0}^t a_z X^z$$

\*Lagrange interpolation



## An Example in $\mathbb{R}$ (Storing)

$$S = (s_1, \dots, s_i, \dots, s_n)$$

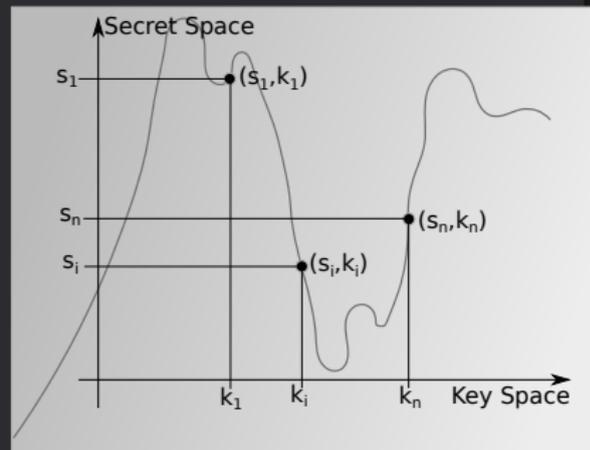
$$K = (k_1, \dots, k_i, \dots, k_n)$$

$$c = \text{store}_K(S)$$

$$c = \Lambda_K^*(S)$$

$$c = f = \sum_{z=0}^t a_z x^z$$

\* Lagrange interpolation

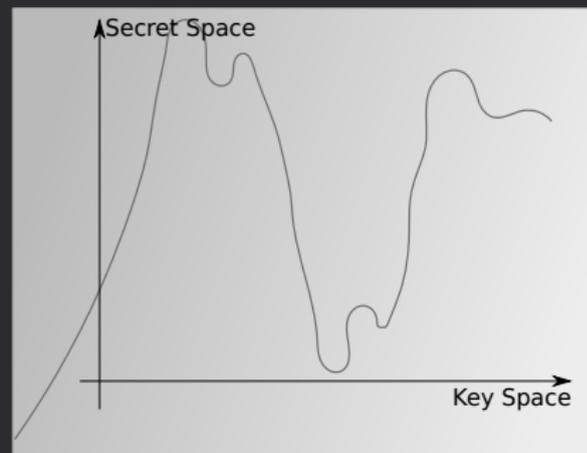


## An Example in $\mathbb{R}^2$ (Retrieving)

$$c = f = \sum_{z=0}^t a_z x^z$$

$$s_i = \text{retrieve}_{k_i}(c)$$

$$s_i = f(k_i) = \sum_{z=0}^t a_z k_i^z$$

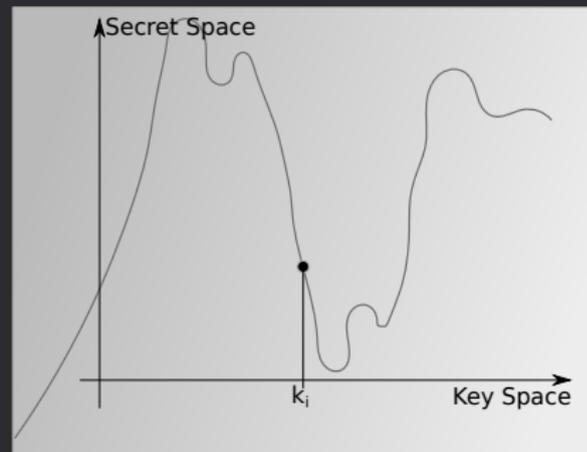


## An Example in $\mathbb{R}^2$ (Retrieving)

$$c = f = \sum_{z=0}^t a_z x^z$$

$$s_i = \text{retrieve}_{k_i}(c)$$

$$s_i = f(k_i) = \sum_{z=0}^t a_z k_i^z$$

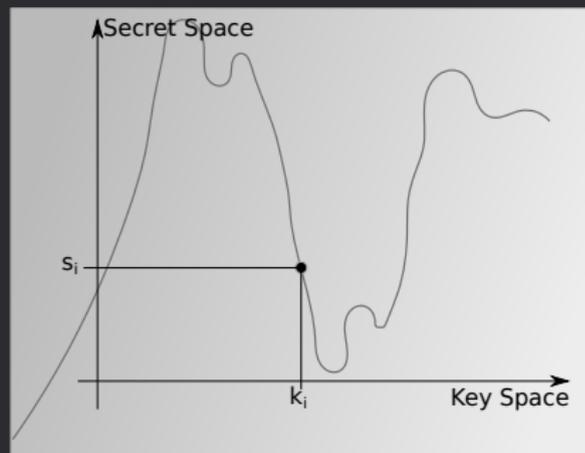


## An Example in $\mathbb{R}^2$ (Retrieving)

$$c = f = \sum_{z=0}^t a_z x^z$$

$$s_i = \text{retrieve}_{k_i}(c)$$

$$s_i = f(k_i) = \sum_{z=0}^t a_z k_i^z$$



## Considerations

Cryptographers do not like  $\mathbb{R}$ ... too chatty... too dense

The secret-storing system hence is created in a Galois field  $\mathbb{F}_p$

---

Problems:

## Considerations

Cryptographers do not like  $\mathbb{R}$ ... too chatty... too dense

The secret-storing system hence is created in a Galois field  $\mathbb{F}_p$

---

### Problems:

- How can the user be free in its choice of  $k$  if the field  $\mathbb{F}_p$  is finite?
- How can the secret space  $|S|$  be reduced / how can  $\mathbb{F}_p$  be made  $2^k$  times greater than  $|S|$ ? ( $k \approx 100$ )

## Considerations

Cryptographers do not like  $\mathbb{R}...$  too chatty... too dense

The secret-storing system hence is created in a Galois field  $\mathbb{F}_p$

---

Problems:

- How can the user be free in its choice of  $k$  if the field  $\mathbb{F}_p$  is finite?
- How can the secret space  $|S|$  be reduced / how can  $\mathbb{F}_p$  be made  $2^k$  times greater than  $|S|$ ? ( $k \approx 100$ )

## Considerations

Cryptographers do not like  $\mathbb{R}$ ... too chatty... too dense

The secret-storing system hence is created in a Galois field  $\mathbb{F}_p$

---

### Problems:

- How can the user be free in its choice of  $k$  if the field  $\mathbb{F}_p$  is finite?
- How can the secret space  $|S|$  be reduced / how can  $\mathbb{F}_p$  be made  $2^k$  times greater than  $|S|$ ? ( $k \approx 100$ )

## Considerations

### Problems:

- How can the user be free in its choice of  $k$  if the field  $\mathbb{F}_p$  is finite?
- How can the secret space  $|S|$  be reduced / how can  $\mathbb{F}_p$  be made  $2^k$  times greater than  $|S|$ ? ( $k \approx 100$ )

### Solution:

Use a collision free mapping function per storage:

$$\kappa_K(k_0) : K \mapsto K', k_0 \in \mathbb{Z}, k'_i \in \mathbb{F}_p \forall k'_i \in K', |K| = |K'|$$

$$\kappa_{k_i}(k_0) : k_i \mapsto k'_i, k_0 \in \mathbb{Z}, k_i \in K, k'_i \in K'$$

The secret  $c$  now consists of the two-tuple  $(f, k_0)$

## Considerations

### Problems:

- How can the user be free in its choice of  $k$  if the field  $\mathbb{F}_p$  is finite?
- How can the secret space  $|S|$  be reduced / how can  $\mathbb{F}_p$  be made  $2^k$  times greater than  $|S|$ ? ( $k \approx 100$ )

### Solution:

Use a collision free mapping function per storage:

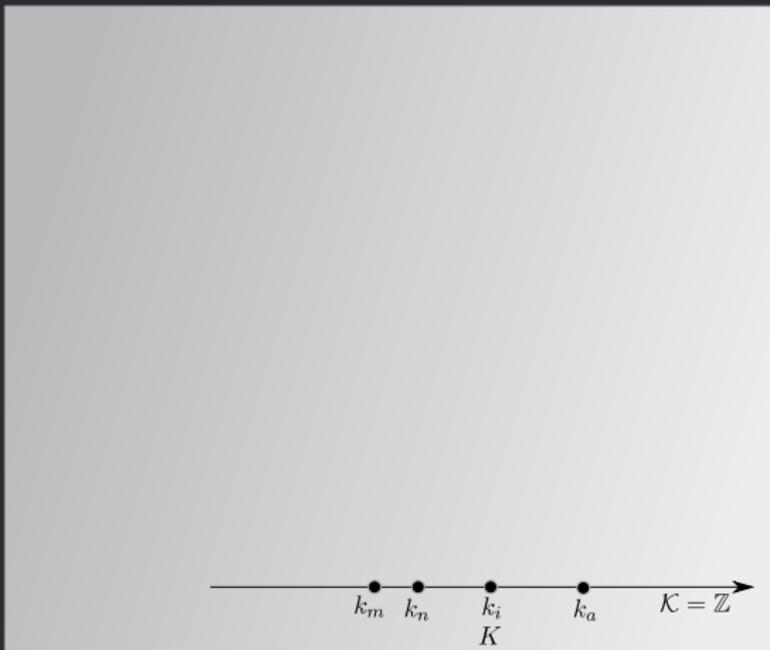
$$\mathcal{K}_K(k_0) : K \mapsto K', k_0 \in \mathbb{Z}, k'_i \in \mathbb{F}_p \forall k'_i \in K', |K| = |K'|$$

$$\mathcal{K}_{k_i}(k_0) : k_i \mapsto k'_i, k_0 \in \mathbb{Z}, k_i \in K, k'_i \in K'$$

The secret  $c$  now consists of the two-tuple  $(f, k_0)$

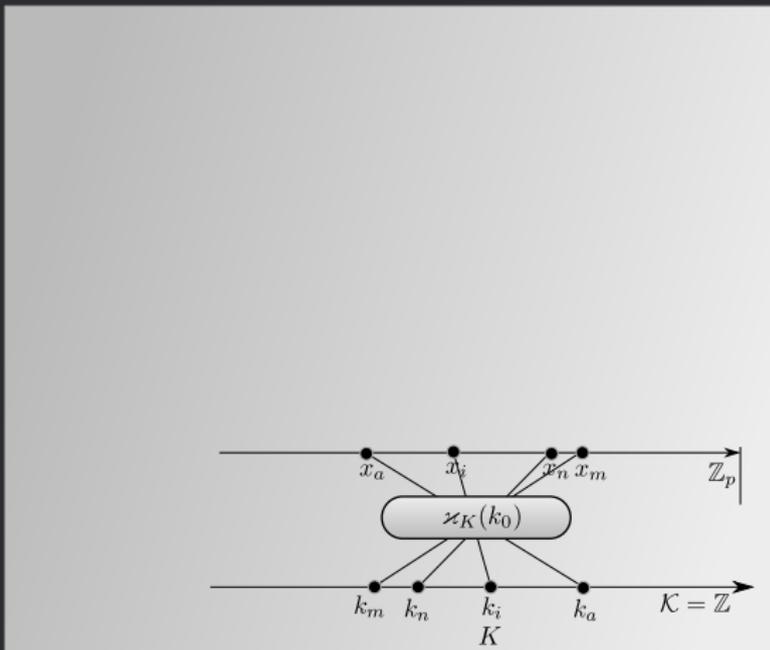
## Considerations: Enhancing the Key Space

Use a collision free mapping function per storage  $\mathcal{K}(k_0)$



## Considerations: Enhancing the Key Space

Use a collision free mapping function per storage  $\varkappa_K(k_0)$



## Considerations

### Problems:

- How can the user be free in its choice of  $k$  if the field  $\mathbb{F}_p$  is finite?
- How can the secret space  $|\mathcal{S}|$  be reduced / how can  $\mathbb{F}_p$  be made  $2^k$  times greater than  $|\mathcal{S}|$ ? ( $k \approx 100$ )

### Solution:

Use a function  $\zeta : \mathbb{Z}_p \mapsto \mathcal{S}$  where  $\zeta^{-1}$  is surjective and easy to calculate:

$$\zeta := s_j \equiv y_j \pmod{|\mathcal{S}|}, \text{ where } y_j = f(k')$$

$$\zeta^{-1} := y_j = s_j + q \times |\mathcal{S}|, q \in_R \mathbb{N}, y_j < p$$

## Considerations

### Problems:

- How can the user be free in its choice of  $k$  if the field  $\mathbb{F}_p$  is finite?
- How can the secret space  $|\mathcal{S}|$  be reduced / how can  $\mathbb{F}_p$  be made  $2^k$  times greater than  $|\mathcal{S}|$ ? ( $k \approx 100$ )

### Solution:

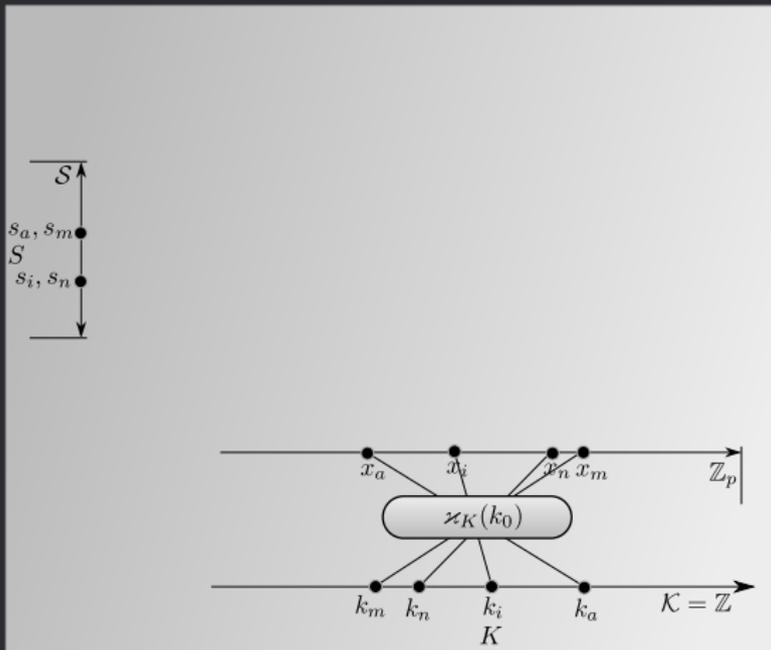
Use a function  $\zeta : \mathbb{Z}_p \mapsto \mathcal{S}$  where  $\zeta^{-1}$  is surjective and easy to calculate:

$$\zeta := s_i \equiv y_i \pmod{|\mathcal{S}|}, \text{ where } y_i = f(k')$$

$$\zeta^{-1} := y_i = s_i + q \times |\mathcal{S}|, q \in_R \mathbb{N}, y_i < p$$

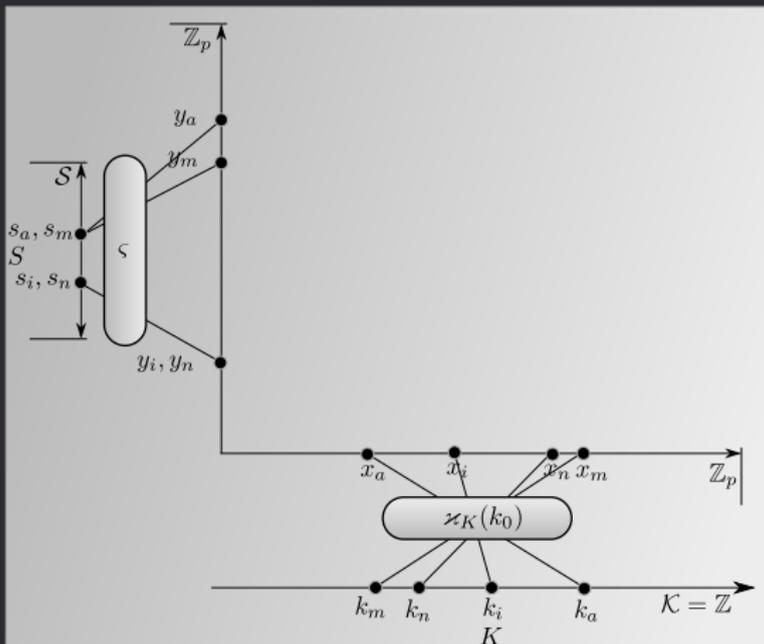
## Considerations: Reducing the Secret Space

Use a function  $\zeta : \mathbb{Z}_p \mapsto \mathcal{S}$



## Considerations: Reducing the Secret Space

Use a function  $\zeta : \mathbb{Z}_p \mapsto \mathcal{S}$









## Considerations

$$S = (s_1, \dots, s_i, \dots, s_n)$$

$$K = (k_1, \dots, k_i, \dots, k_n)$$

$$c = \text{store}_K(S)$$

$$K' = \varkappa_K(k_0)$$

$$S' = \varsigma^{-1}(S)$$

$$f = \Lambda_{K'}(S')$$

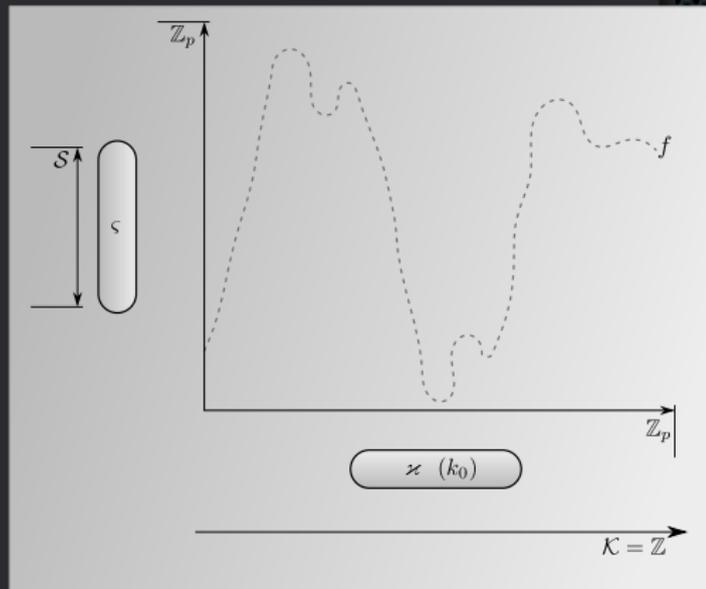
$$c = (f, k_0)$$

$$s_i = \text{retrieve}_{k_i}(c)$$

$$k'_i = \varkappa_{k_i}(k_0)$$

$$s'_i = f(k'_i)$$

$$s_i = \varsigma(s'_i)$$



## Considerations

$$S = (s_1, \dots, s_i, \dots, s_n)$$

$$K = (k_1, \dots, k_i, \dots, k_n)$$

$$c = \text{store}_{k_i}(S)$$

$$K' = \mathcal{K}_{k_i}(k_0)$$

$$S' = \mathcal{F}(S)$$

$$f = \Lambda_{K'}(S')$$

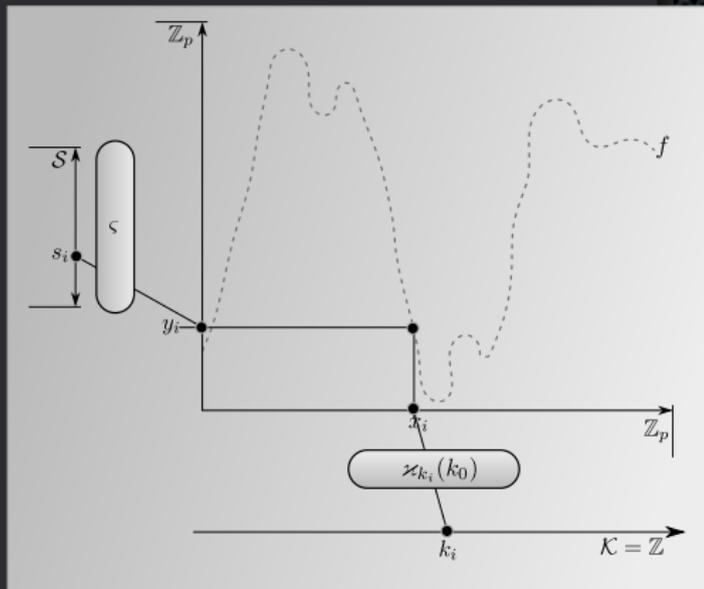
$$c = (f, k_0)$$

$$s_i = \text{retrieve}_{k_i}(c)$$

$$k'_i = \mathcal{K}_{k_i}(k_0)$$

$$s'_i = f(k'_i)$$

$$s_i = \mathcal{S}(s'_i)$$



## How to Use It For the 20-Credentials

### A simple example (it works, but...)

- ...use consonants as keys for the 'Drop' credentials
- ...use vowels as keys for 'Accept' credentials
- ...use some other keys for some credentials
- ...store them in the secret-storage system

```

924f61661a3472da74307a35f2c8d22e07e84a4d ○
cbf019b764b9477080c5a9a748a2911a5fa6d614 ○
fc0ccd6641d45ef2efd926c3a6f7f3ac268e9e3 ✓
a29965fbb2954c8a66d856e8eb891bce5f49dacf ✓
3a710d2a84f856bc4e1c0bbb93ca517893c48691 ○
e1a5b5d17e51d56f0d6fc060968ff238afb9b32 ○
cbf019b764b9477080caa9a748a2911a5fa6d614 ○
a29965fbb2954c8a66d8b6e8eb891bce5f49dacf ○
22eb60281c37e6611e85e7a432a45c8f3525749 ●
924f61661a34d2da74307a35f2c8d22e07e84a4d ○
f5abae583297649047c13be2c54bcbfb3268f0f3 ○
b1aa98ad3a02ffe896c49687300d8644f50fd88 ○
2789a4eb84e43e4d5b60d87b3d1edec02d4c449e ○
2789a4eb84e43e4d5b60d87b3d1edec02d4c449e ○
fc0ccd6641d45ef2efd946c3a6f7f3ac268e9e3 ●
e1a5b5d17e51d56f0d6fc060e68ff238afb9b32 ○
fd8b823d965947fc7d9f470907ca18ed68243557 ○
f5abae583297649547c13be2c54bcbfb3268f0f3 ✓
3a710d2a84f856bcce1c0bbb93ca517893c48691 ✓
b1aa98ad3a02ffe896c49687300d8644f50fd88 ✓
22eb60281c37e6611e85e7a432a45c8f3525749 ○
fd8b823d965947fc7d9f470907ca18ed68243557 ○

```

## How to Use It For the 20-Credentials

### A simple example (it works, but...)

- ...use consonants as keys for the 'Drop' credentials
- ...use vowels as keys for 'Accept' credentials
- ...use some other keys for some credentials
- ...store them in the secret-storage system

```

924f61661a3472da74307a35f2c8d22e07e84a4d ○ ○ c
cbf819b764b9477080c5a9a748a2911a5fa6d614 ○ ○ c
fc8ccd6641d45ef2efdd926c3a6f7f3ac268e9e3 ✓ ✓ d
a29965fbb2954c8a66d856e8eb891bce5f49dacf ✓ ✓ d
3a718d2a84f856bcc4e1c0bbb93ca517893c48691 ○ ○ f
e1a5b5d17e51d56f0d6fc060968ff238afb9b32 ○ ○ f
cbf819b764b9477080caa9a748a2911a5fa6d614 ○ ○ f
a29965fbb2954c8a66d8b6e8eb891bce5f49dacf ○ ○ f
● 22eb602811c37e6611e85e7a432a45c8f3525749 ○ ○ g
924f61661a34d2da74307a35f2c8d22e07e84a4d ○ ○ g
f5abae503297649047c13be2c54bcbfb3260f0f3 ○ ○ h
b1aa98ad3a02ffe896c49687308d8644f58fdd88 ○ ○ h
2789a4eb84e43e4d5b60d87b3d1edec02d4c449e ○ ○ i
2789a4eb84e43e4d5b60d87b3d1edec02d4c449e ○ ○ i
● fc8ccd6641d45ef2efdd946c3a6f7f3ac268e9e3 ○ ○ j
e1a5b5d17e51d56f0d6fc060e68ff238afb9b32 ○ ○ j
fd8b823d965947fc7d9f478987ca18ed60243557 ○ ○ k
f5abae503297649547c13be2c54bcbfb3260f0f3 ○ ○ k
3a718d2a84f856bcc1c0bbb93ca517893c48691 ✓ ✓ l
b1aa98ad3a02ffe896c49687308d8644f58fdd88 ○ ○ l
22eb602841c37e6611e85e7a432a45c8f3525749 ○ ○ m
fd8b823d985947fc7d9f478987ca18ed60243557 ○ ○ m

```

## How to Use It For the 20-Credentials

### A simple example (it works, but...)

- ...use consonants as keys for the 'Drop' credentials
- ...use vowels as keys for 'Accept' credentials
- ...use some other keys for some credentials
- ...store them in the secret-storage system

```

924f61661a3472da74387a35f2c8d22e87e84a4d ○
cbf019b764b9477080c5a9a748a2911a5fa6d614 ○
fc8c06641d45ef2efdd926c3a6f7f3ac268e9a3 ○
a29965fbb2954c8a66d856e8eb891bce5f49dacf ○
3a718d2a84f856bc4e1c0bb093ca517893c48691 ○
e1a5b5d17e51d56f06fc869968ff238a7ba9b32 ○
cbf019b764b9477080c5a9a748a2911a5fa6d614 ○
a29965fbb2954c8a66d856e8eb891bce5f49dacf ○
22eb602841c37e6611e85e7a432a45c8f3525749 ●
924f61661a3472da74387a35f2c8d22e87e84a4d ○
f5abae503297649047c13be2c54bcfb3260f0f3 ○
b1aa98ad3a02ffe896c49687300d8644f50fd088 ○
2789a4eb084e43e4d5b6087b3d1edec8204c449e ○
2789a4eb084e43e4d5b6087b3d1edec8204c449e ○
fc8c06641d45ef2efdd926c3a6f7f3ac268e9a3 ○
e1a5b5d17e51d56f06fc869968ff238a7ba9b32 ○
f080230965947fc7d9f478907ca18ed60243557 ○
f5abae503297649047c13be2c54bcfb3260f0f3 ○
3a718d2a84f856bc4e1c0bb093ca517893c48691 ○
b1aa98ad3a02ffe896c49687300d8644f50fd088 ○
22eb602841c37e6611e85e7a432a45c8f3525749 ○
f080230965947fc7d9f478907ca18ed60243557 ○

```

c  
e  
i  
o  
u  
f  
h  
g  
k  
j  
r  
l  
m  
n  
p  
q  
a  
o  
t  
s

## How to Use It For the 20-Credentials

### A simple example (it works, but...)

- ...use consonants as keys for the 'Drop' credentials
- ...use vowels as keys for 'Accept' credentials
- ...use some other keys for some credentials
- ...store them in the secret-storage system

```

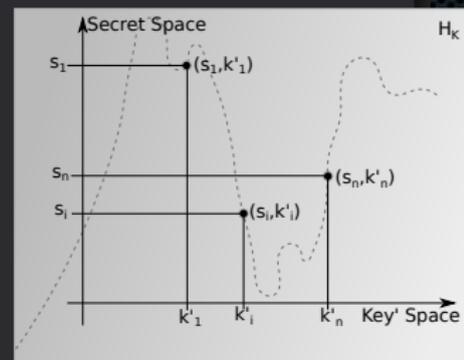
924f61661a3472da74387a35f2c8d22e87e84a4d obad
cbf019b764b9477080c5a9a748a2911a5fa6d614 c
fc8c06641d45ef2ef0926c3a6f7f3ac268e9a3 e
a29965fbb2954c8a66d8556e8e891bce5f49dacf i
3a718d2a84f856bc4e1c0bb93ca517893c48691 d
e1a5b5d17e51d56f06fc869968ff238a7ba9b32 f
cbf019b764b9477080c5a9a748a2911a5fa6d614 h
a29965fbb2954c8a66d8556e8e891bce5f49dacf g
22eb602841c37e6611e85e7a432a45c8f3525749
924f61661a3472da74387a35f2c8d22e87e84a4d
f5abae503297649047c13be2c54bcfb3260f0f3 j
b1aa98ad3a02ffe896c4968730008644f50fd088 r
27894e084e43e4d5b6087b3d1edec8204c449e u
27894e084e43e4d5b6087b3d1edec8204c449e m
fc8c06641d45ef2ef0926c3a6f7f3ac268e9a3 n
e1a5b5d17e51d56f06fc869968ff238a7ba9b32 p
f080230965947fc7d9f478907ca18ed60243557 q
f5abae503297649047c13be2c54bcfb3260f0f3 good
3a718d2a84f856bc4e1c0bb93ca517893c48691 a
b1aa98ad3a02ffe896c4968730008644f50fd088 o
22eb602841c37e6611e85e7a432a45c8f3525749 t
f080230965947fc7d9f478907ca18ed60243557 s

```

## How to Use It For the 20-Credentials

### A simple example (it works, but...)

- ...use consonants as keys for the 'Drop' credentials
- ...use vowels as keys for 'Accept' credentials
- ...use some other keys for some credentials
- ...store them in the secret-storage system



The secret-storing system brings other very desirable features:

Typoo Resistance via Secret-Storing System | **The OR-Function**

Either  $key_i$  OR  $key_n$  unveils the single secret

$$s_i = \text{retrieve}_{k_i}(c)$$

$$s_n = \text{retrieve}_{k_n}(c)$$

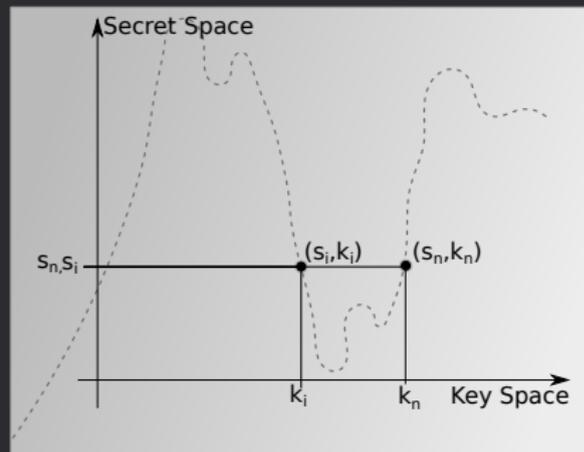
$$\text{verify}(s_i = s_n)$$

The secret-storing system brings other very desirable features:

## Typoo Resistance via Secret-Storing System | The OR-Function

Either  $key_i$  OR  $key_n$  unveils the single secret

$s_i = \text{retrieve}_{k_i}(c)$   
 $s_n = \text{retrieve}_{k_n}(c)$   
 $\text{verify}(s_i = s_n)$



## Key-Hinting via Secret-Storing System | The AND Function

Only  $key_i$  AND  $key_n$  unveil the single secret:

$k_a$  "HintMeWith...:"

$k_b$  "GoodCredentials"

$k_c$  "BadCredentials"

$k_d$   $s_a \oplus s_b$

$k_e$   $s_a \oplus s_c$

$s_a$  "098z71adf4383498"

$s_b$  "aer9393fads932sv3"

$s_c$  "598nnja2devm24v3a"

$s_d$  "vowel"

$s_e$  "consonant"

---

This still is secure as long as the keys carries some entropy!

It is definitely more secure than "What is your mother's maiden name" used nowadays!

And this one for free:

## Secret-Sharing via Secret-Storing System | The THRESHOLD Function

Only  $n$  out of  $m$  keys unveil the single secret...

$k_a$  "Chief-1"

$k_b$  "Chief-2"

$k_c$  "Chief-3"

$k_x$   $s_a \oplus s_b$

$k_y$   $s_a \oplus s_c$

$k_z$   $s_b \oplus s_c$

$s_a$  "098z71adf4383498"

$s_b$  "1gfnasdrhhsadfn"

$s_c$  "fgjn439f3j22tj93"

$s_x$  "We are bankrupt"

$s_y$  "We are bankrupt"

$s_z$  "We are bankrupt"

## Randomized Secret-Storing System

If some sample positions  $rx_j \notin K$  are chosen arbitrary with arbitrary sample values  $ry_j$  ( $R = m\text{-tuple}(rx, ry)$ ), the system changes from a deterministic secret-storing system into a randomized secrets-storing System, where two stores containing the same  $(K, S)$  'look' completely different.

### Definition

A randomized secret-storing system of order  $n$ ,

$$\Gamma = (\mathcal{S}, \mathcal{K}, \mathcal{R}, \mathcal{C}, \text{randomStore}, \text{retrieve}),$$

consists of a secret space  $\mathcal{S}$ , a key space  $\mathcal{K}$ , a randomization space  $\mathcal{R}$ , a storage space  $\mathcal{C}$ , and two functions `randomStore` and `retrieve` with properties as introduced above.

## Randomized Secret-Storing System

If some sample positions  $rx_j \notin K$  are chosen arbitrary with arbitrary sample values  $ry_j$  ( $R = m\text{-tuple}(rx, ry)$ ), the system changes from a deterministic secret-storing system into a randomized secrets-storing System, where two stores containing the same  $(K, S)$  'look' completely different.

### Definition

A *randomized secret-storing system* of order  $n$ ,

$$\Gamma = (\mathcal{S}, \mathcal{K}, \mathcal{R}, \mathcal{C}, \text{randomStore}, \text{retrieve}),$$

consists of a secret space  $\mathcal{S}$ , a key space  $\mathcal{K}$ , a randomization space  $\mathcal{R}$ , a storage space  $\mathcal{C}$ , and two functions `randomStore` and `retrieve` with properties as introduced above.

## The Thing is a Beast

### Ring-homomorphism

Every operation  $(+, *)$  can be applied using multiple secret-stores as operands. The same operations will then be applied on all secrets of the same key.

This is exactly the most desired feature for a secure\* and private\* web-service\* in the cloud\* on the internet\* .

A CPU in the cloud can now do calculations using secret-stores, hence not knowing the operands nor the result.

---

\*Bingo ;-)

## The Thing is a Beast

### Ring-homomorphism

Every operation  $(+, *)$  can be applied using multiple secret-stores as operands. The same operations will then be applied on all secrets of the same key.

This is exactly the most desired feature for a secure\* and private\* web-service\* in the cloud\* on the internet\* .

A CPU in the cloud can now do calculations using secret-stores, hence not knowing the operands nor the result.

---

\*Bingo ;-)

## Discussion

This is left as an exercise for the reader...

...One more Thing...

The Asymmetric variant of the secret-storing system with the same properties as shown above.

Please feel Free to Break the Secret-Storing System!

As it is a rather easy concept (even for a young math-student), it is easy to understand the system and thus to look for the show-stopper.... So, if you ever... Please let me know too!

# Happy Hacking!

