Cry – Project 1 (Project Plan): Presentation

2017-01-30

Daniel Dunning, Michael Degraw, Vu Phan

Don't TEX and drive.

Donald

Section 1

Motivation

end-user

- Problem: An end-user wants to send and receive secure messages with other end-users.
- Solution: `Cry` lets end-users establish secure communication via built-in cryptosystems.

cryptographer

- Problem: As new technologies emerge, and as technology improves, new cryptosystems need to be developed.
- Solution: `Cry` allows cryptographers to easily prototype and benchmark their new cryptosystems.

Section 2

Overview

Subsection 1

Example Use Case

Situation

- *Alice* (sender) wants to confidentially send a message to *Bob* (receiver).
- *Eve* (eavesdropper) wants to know that message.

Procedure

- *Bob* generates his keys:

```
$ cry getkeys --cryptosystem=rsa
The public & private keys are 825 & 637 (took 1 second).
```

- *Alice* (sender)

```
$ cry encrypt --cryptosystem=rsa \
> --public-key=825 --plaintext=4692301804
The ciphertext is 1110003333 (took 1 second).
```

Procedure

- *Bob* (receiver)

```
$ cry decrypt --cryptosystem=rsa \
> --private-key=637 --ciphertext=1110003333
The plaintext is 4692301804 (took 1 second).
```

- *Eve* (eavesdropper)

```
$ cry eavesdrop --cryptosystem=rsa \
> --public-key=825 --ciphertext=1110003333
The plaintext is 4692301804 (took 1 century).
```

Subsection 2

Functionality

In the previous example:

- `Cry` is the cryptoframework.
- RSA is a cryptosystem implemented in `Cry`.
- The key-generation, encryption, decryption, and eavesdropping algorithms are specific to RSA.

In general, with Cry:

- an end-user can use an implemented cryptosystem to confidentially send and receive messages with others.
- a cryptographer can:
  - prototype her own cryptosystems where the cryptographic algorithms are either newly defined or reused from different existing cryptosystems.
  - test her cryptosystems for security and performance.

Section 3

Expectation

Usability

- For end-users: `Cry` enables end-users to establish secure communication via simple command-line options.
- For cryptographers:
  - Being a modular object-oriented framework, `Cry` lets cryptographers easily define new cryptosystems.
  - `Cry` also reports the security measures of these cryptosystems.

in Comparison to an Existing Framework

- `Charm` is an exisiting cryptoframework implemented in Python and C.
- `Charm` has advanced functionalities that we will be unable to provide in `Cry` due to time constraints.
- `Cry` will have a similar base for sandboxing and prototyping cryptosystems.

Section 4

Conclusion

last and least

Source: `https://github.com/vuphan314/cry`

To `Cry`, or not to `Cry`?

Will