Secure Publish Subscribe Compute System

Anonymous Author(s)

ABSTRACT

Your abstract should go here. You will also need to upload a plain-text abstract into the web submission form.

CCS CONCEPTS

• **Security and privacy** → Use https://dl.acm.org/ccs.cfm to generate actual concepts section for your paper;

KEYWORDS

template, formatting, pickling

1 INTRODUCTION

2 RELATED WORK

3 PROTOCOL

* We minimize the communication with `Garbler`, using clever tricks, e.g., we don't need to send labels to `Garbler` for circuit garbling instead they generate it independently using shared seed.

* publishers and subscribers only talk to the broker. The communication between publishers and subscriber is done through `Broker` using end-to-end encryption. This ensures that

* Does the subscriber seed needs to be computation specific?

* We use ratcheting for forward security if seed is compromised.

* We assume PKI.

Intuition.

3.1 Basic Protocol

Do we have to generate seed and ratchet key? Can't we use the seed as a ratchet key?

Forward Security.

- Generate a truly random key K_0.
- Generate, using KDF with key K_0, a pseudorandom seed s_0 and a pseudorandom key for the ratchet round 1. Seed s_0 is used to generate pseudorandom strings during ratchet round 0.
- At round r, using KDF with key K_r, generate a pseudorandom seed s_r and key for ratchet round $r+1$. Seed s_r is used to generate pseudorandom strings during ratchet round r.

Garbled Circuit XOR Compatibility. * What if one publisher doesn't send wire labels. After timeout the broker can inform the `Garbler` and it will use zero for such values in the circuit.

4 SYSTEM

Extending libgarble.

Identity Gates for FreeXOR Compatibility.

5 EVALUATION

6 CONCLUSION

Figure 1: Basic Protocol

Initialization.

- Each new publisher sends `Broker` a policy specifying allowed computations on its data and generates a truly random seed s and send it to `Garbler`.
- All publishers and subscribers establish an authenticated encrypted channel with `Broker` and through `Broker` with `Garbler`.

Subscription.

- To subscribe computation C, subscriber sends a subscription request containing C to `Broker` and requests an output masking seed from `Garbler`.
- `Garbler` sends a truly random seed s' for computation C; generating a new seed if this is the first subscription for computation C.

Publication.

- To publish kth value, publisher generates two pseudorandom wire labels, w_0 and w_1, using seed s, for each bit of the value. w_0 is ith and w_1 is $(i+1)$th numbers in pseudorandom sequence generated using seed s; $2kL \leq i < 2(k+1)L$, L being the bit-length of a value.

Computation.

- Once `Broker` has wire labels for all publishers' inputs required for computation C, it requests `Garbler` to garble a circuit for C.
- `Garbler` independently generates input wire labels using seed s from each publisher contributing input and an output mask m using seed s' for each output bit.
- `Garbler` generates garbled circuit for $M \circ C(\cdot)$, composition of masking function $M : XOR$ and computation C, and sends it to `Broker`.
- `Broker` evaluates the garbled circuit using wire labels sent by publishers, obtains masked output $o \oplus m$, and send $o \oplus m$ to all subscribers of computation C.
- Subscribers generate the mask m using the seed s' and unmask the output o.