Oblivious Pseudo-Random Functions (OPRFs)

An example of newer crypto primitives
(sort-of extending the idea of hash functions)

https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-voprf

CS7NS5/CSU44032

stephen.farrell@cs.tcd.ie

https://down.dsg.cs.tcd.ie/cs7053/

https://github.com/sftcd/cs7053

Introduction

- This a new twist on some relatively old ideas but now with some possibly topical applications
- Yay! Basing on old ideas (from 1992 in this case) is somewhat re-assuring WRT patents on the basic ideas
- Yay! New topical applications – justify the (non-trivial) effort in mapping from a peer-reviewed cryptographic publication to an implementable API/protocol/application
- But – don’t get carried away, not all new things end up important

Discrete Log Equivalence: DLEQ

- Chaum, D., Pedersen, T.P. (1993). Wallet Databases with Observers. CRYPTO’ 92, section 3.2 (https://doi.org/10.1007/3-540-48071-4_7)
- In a prime order group (order q), if we have secret scalar k and group elements such that $B = A^k$ and $D = C^k$, a “prover” can prove this to a “verifier” without exposing k
- Everyone knows q, A, B, C, D
- Prover chooses random s, computes A^s and C^s; Prover sends A^s and C^s to verifier
- Verifier chooses random challenge c and sends that to prover
- Prover sends back “the proof”: $r = s + c.k$
- Verifier accepts proof if $A^r == a.B^c$ and $C^r == b.D^c$

Additive description of DLEQ

- Changing description of original DLEQ to better match VOPRF spec...
  - It’s really the same but just looks different:-)
- In a prime order group (order p), if we have secret scalar k and group elements such that B = k.A and D = k.C, a “prover” can prove this to a “verifier” without exposing k
- Everyone knows p, A, B, C, D
- Prover chooses random s, computes a = s.A and b = s.C
- Prover sends a and b to verifier
- Verifier chooses random challenge c and sends that to prover
- Prover sends back “the proof”: r = s + c.k
- Verifier accepts proof if r.A == a + c.B and r.C == b + cD
- Note: I think the above is correct but we’re at the limits of my crypto skills:-)

Pseudo Random Function (PRF)

A function F(k,_) is pseudorandom if the keyed function K(_) = F(k, _) is indistinguishable from a randomly sampled function acting on the same domain and range as K()

Cryptographers care about the above as it affects what can be proven, but for us, it’s just a matter of definition

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 25 August 2023

A. Davidson
Brave Software
A. Faz-Hernandez
N. Sullivan
C. A. Wood
Cloudflare, Inc.

21 February 2023

Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups

draft-irtf-cfrg-voprf-21

OPRF, VOPRF, POPRF

An Oblivious PRF (OPRF) is a two-party protocol between a server and a client, where the server holds a PRF key k and the client holds some input x. The protocol allows both parties to cooperate in computing F(k, x) such that the client learns F(k, x) without learning anything about k; and the server does not learn anything about x or F(k, x).

A Verifiable OPRF (VOPRF) is an OPRF wherein the server also proves to the client that F(k, x) was produced by the key k corresponding to the server's public key the client knows.

A partially oblivious PRF (POPRF) is a VOPRF with an additional public input y, based on F(k, x, y).

Proof/Verify Functions

- The draft...https://datatracker.ietf.org/doc/html/draft-irtf-cfrg-voprf
- ... builds on DLEQ so that we can prove the equivalence for a set of m+1 values at once, not just two (without affecting the size of the proof)
- ... uses additive notation e.g. B = k.A rather than B = A^k but don't let that bother you (they're groups, and in practice elliptic curve groups)
- ... replaces exchange of random numbers with hashes/transcripts based on input values and context-distinguishing strings (e.g. the string "Challenge") – the lengths of things are explicitly part of such transcripts – this makes the DLEQ scheme non-interactive ('cause we don't need a client to choose the challenge 'c')
- Section 2.2 isn't entirely obvious, but if you work it out, you'll conclude that believing the DLEQ result means beliving this too
- The draft is basically dealing with the usual elliptic curve groups (based on 25519, p256 etc)

Protocols

- The spec defines OPRF, VOPRF and POPRF variant protocols, we'll only look at the last (it does what the others do too)
- In these protocols, the client is a verifier, the server is the prover, but there may be other verifiers later (recall the DLEQ scheme is non-interactive)
- Basic idea is client "blinds" it's input, sends blinded input to server which then uses the DLEQ trick (that's the OPRF version) – blinding can be as simple as multiplying by a random scalar; the client can "unblind" (or "Finalize()") results received from the server
- The server's secret (skS) in the VOPRF or POPRF setting is an asymmetric private value, such that the public value (pkS) allows the client to check the PRF output; the client of course needs to get to know the public value somehow
  - VOPRF plays the DLEQ game with A = the group identity, B = pkS and C[0] = the blinded-input and with C being a 1-element list
- The POPRF version additionally has a public input ("info") known to both client and server that is also fed into the protocol ("tweaking" skS)

POPRF Protocol

```
  Client(input, pkS, info) <---- pkS ------  Server(skS, pkS, info)
-------------------------------------------------------------------
blind, blindedElement, tweakedKey = Blind(input, info, pkS)

                           blindedElement
                             ---------->

       evaluatedElement, proof = BlindEvaluate(skS, blindedElement,
                                               info)

                       evaluatedElement, proof
                             <----------

output = Finalize(input, blind, evaluatedElement,
                  blindedElement, proof, info, tweakedKey)

Figure 3: POPRF protocol overview with additional public input
```

An application

- There are applications of such functions related to secret sharing and PAKEs but perhaps the main motivating one is...
  ... as an alternative to CAPTCHAs
- PrivacyPass is such a technology: idea is to acquire a set of tokens from an issuing server (prover) that can be presented to a verifier, e.g. as evidence there's a human behind the keyboard, without the verifier being able to correlate token usage and without the client being able to cheat
  - https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-protocol
  - Oddly, the privacypass protocol may make use of VOPRFs in a context where the verifier knows the issuer's secret (see section 5.4 of the above draft)

PrivacyPass

- Client is browser browsing that hits an...
- Origin web server that challenges the client who uses an...
- Attester who checks the client passes a CAPTCHA-like thing once, allowing the...
- Issuer to issue e.g. a batch of tokens...
- One of which the client sends to the origin.
- Purpose of the game: origin can't detect same client returning based on tokens used
- Note: Origin might select which issuer(s) it prefers, but there're a few potential deployment models

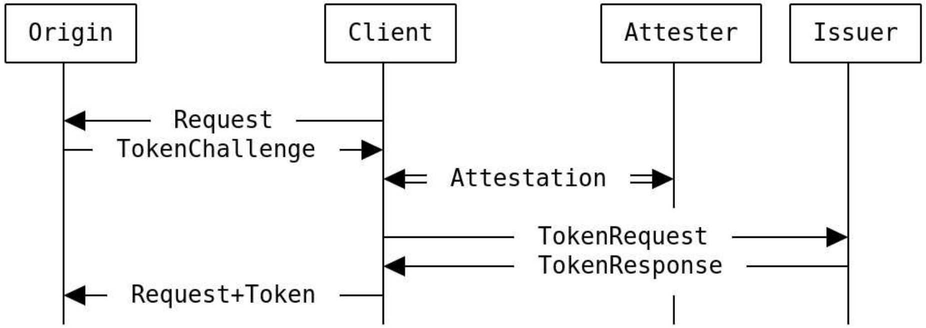

Figure 1: Issuance Overview

Conclusion

- Relatively usable new(er) cryptographic primitives are being defined, often based on earlier more theoretic work
- Those with broad applicability that get used for popular applications may become more important; not many will clear that hurdle (IMO)
- Some of these are non-trivial to understand and use well – that’s a downside: it’s all very well to be super-clever, but (e.g. as engineers) we don’t want to deal with tons of new, less well-known things constantly turning up, as they could increase risk (newer, buggy code; more attack surface; easily mis-understood; more centralising, ...)
- But, our currently “core” cryptgraphic functions, (confidentiality, integrity, hashes, KDFs, PRNGs, etc.) might well be expanded in future (or might not)

More info

- More crypto background: see the references in https://datatracker.ietf.org/doc/draft-irtf-cfrg-voprf/
- More on PrivacyPass: see documents listed at https://datatracker.ietf.org/wg/privacypass