Guaranteeing the integrity of a register

Philip Potter, 13 October 2015 - GaaP, Transformation

Last month we blogged about registers: authoritative lists you can trust. To recap, a register is a canonical source of data, with a clearly-defined custodian. You should be able to trust the integrity of the data that you get from a register.

If you get a record of a company from Companies House, you should be certain that nobody has tampered with the record. Further, you should be able to show it to someone else and demonstrate to them that this is an official record. Similarly, if a restaurant displays a food hygiene rating, you should be able to verify that this rating is genuinely what the Food Standards Agency has recorded for this restaurant. We are exploring mechanisms to achieve this through a digital proof of authenticity.

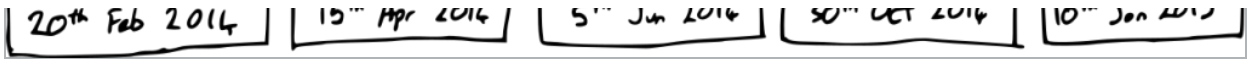
A digital proof of authenticity allows you to:

- check that a record truly was created by the correct authority
- ensure that a registrar never goes back on their word
- copy a register and validate the integrity of its whole contents

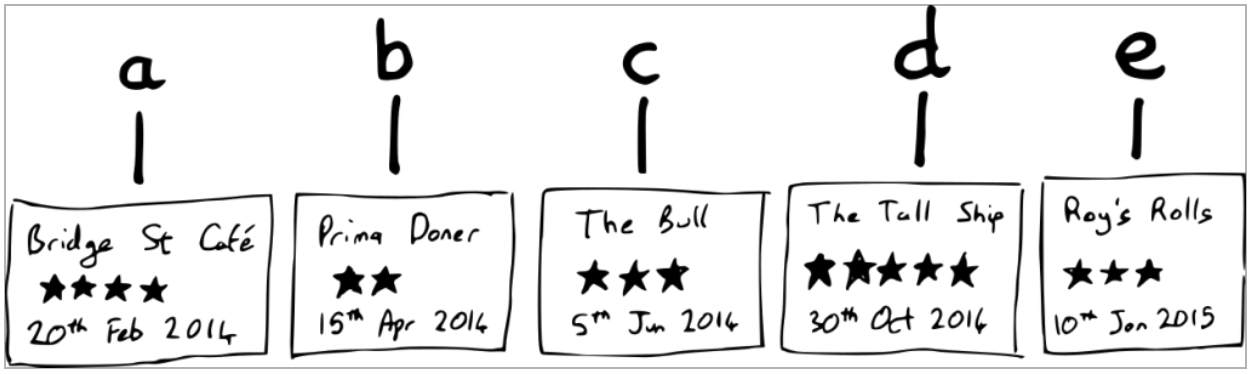
There are a number of ways of achieving this but one we have been exploring is based around Google's Certificate Transparency project. At its heart, Certificate Transparency depends on the creation of a digitally signed append-only log. The entries in the log are hashed together in a Merkle tree and the tree is signed. The registrar can append to the log by issuing a new signature. Consumers can request proof that a single entry appears in a particular log. Consumers can also request proof that the registrar has not rewritten history which the registrar can easily provide.

A worked example

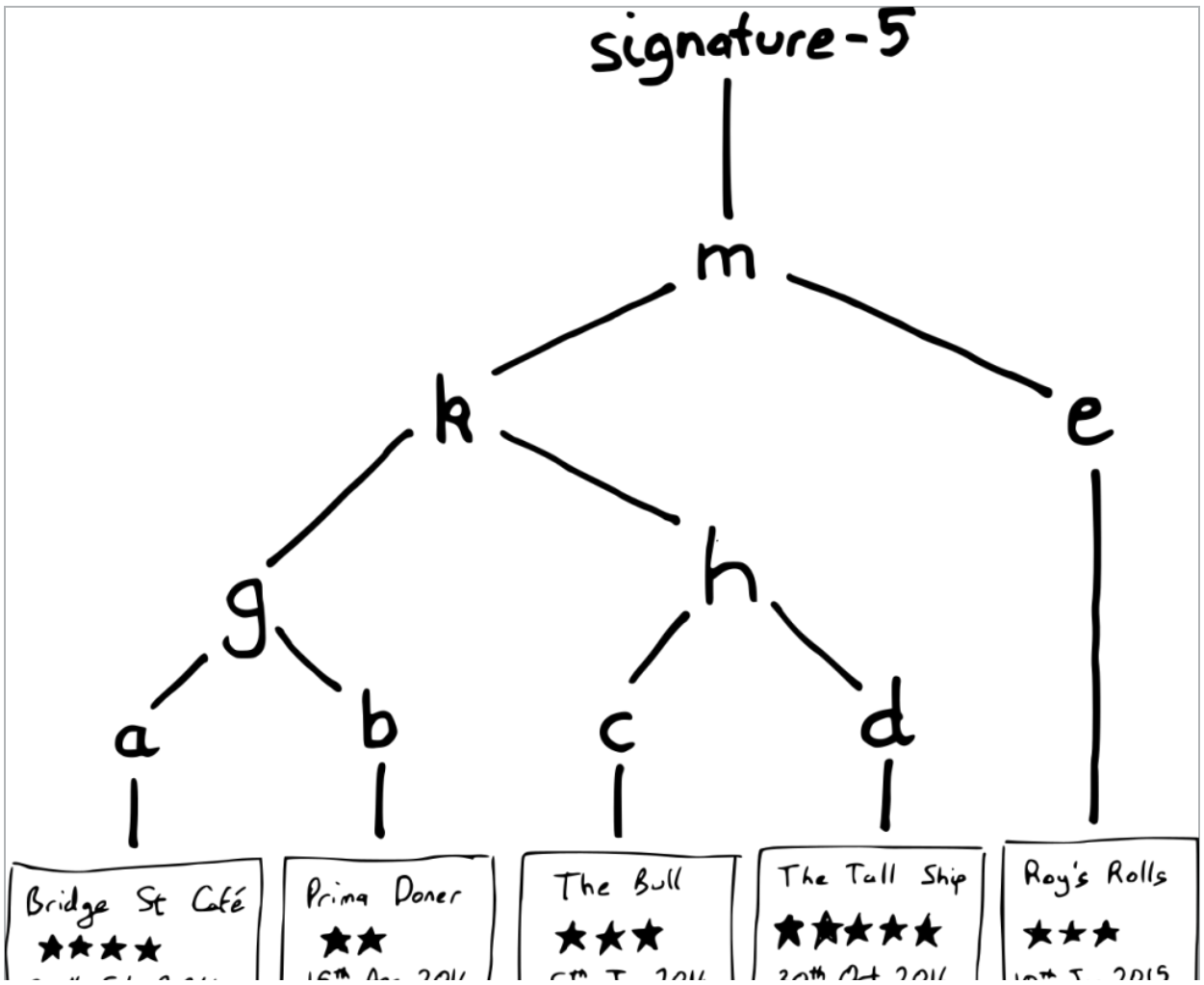
Suppose that we are building a register of restaurant inspections. First, we collect some data:

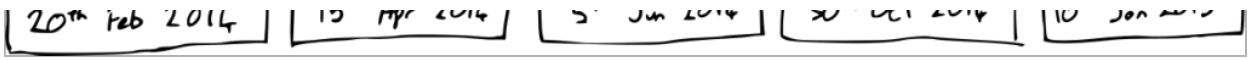


This represents five inspection reports of particular establishments on particular dates. Then, we compute secure hashes of these inspections, labelled here **a**, **b**, **c**, **d**, and **e**:



We then combine these hashes into a Merkle tree, where each tree node is a secure hash of the two elements below it. We compute the hash of **a** and **b** combined, and call it **g**; we compute the hash of **c** and **d**, and call it **h**. We continue this process until we have built a single complete tree with a single head node **m**. Finally, we sign the tree head:

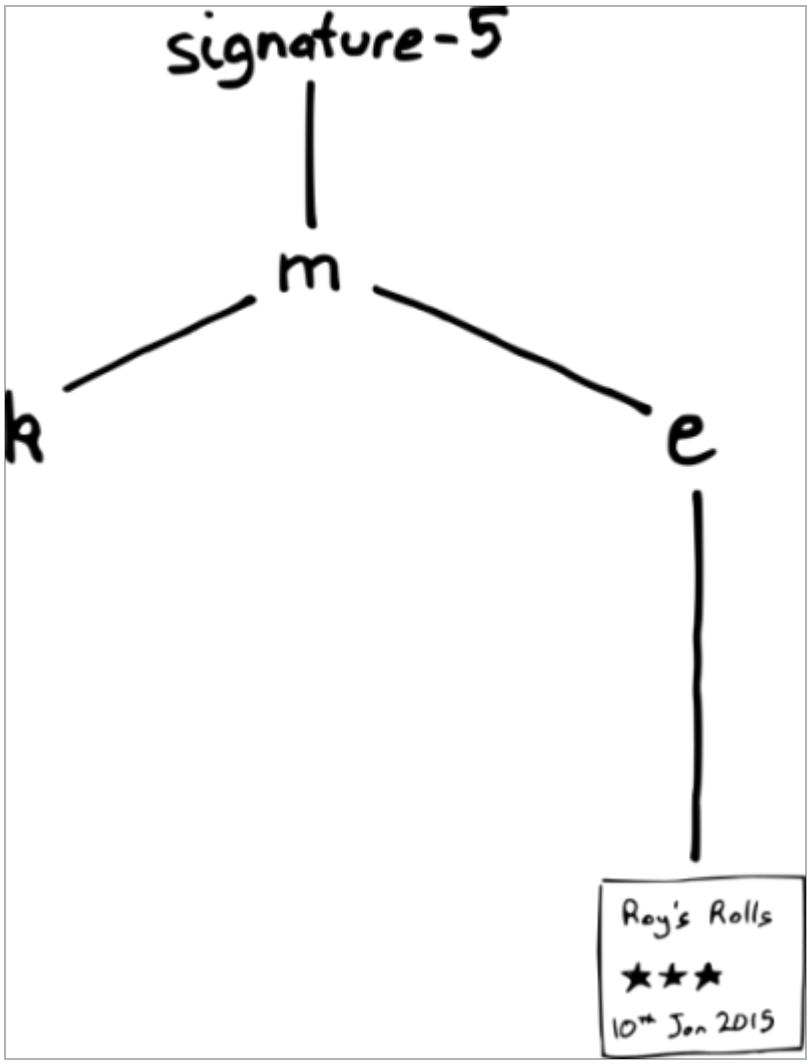




Creating a digital proof

Roy's Rolls wants to display its inspection report on its premises. It also wants to present a digital proof of authenticity of the report, so it asks the register to provide one. To produce this proof, it's worth asking: what is the minimum amount of information you need to validate that the signed tree contains the inspection report of Roy's Rolls? You shouldn't need to download all reports from the register and the whole corresponding Merkle tree in order to validate the digital proof for a single inspection.

If you are given the inspection report, you can compute its hash, **e**. If the register also tells you the value of **k**, you can hash **k** and **e** together to compute **m**. You can then check if the signature validly signs **m**. (You can take the value of **k** on trust because of hash function pre-image resistance.) This means you can ignore most of the tree when checking a single inspection report:



In total, a digital proof of authenticity for the Roy's Rolls inspection report consists

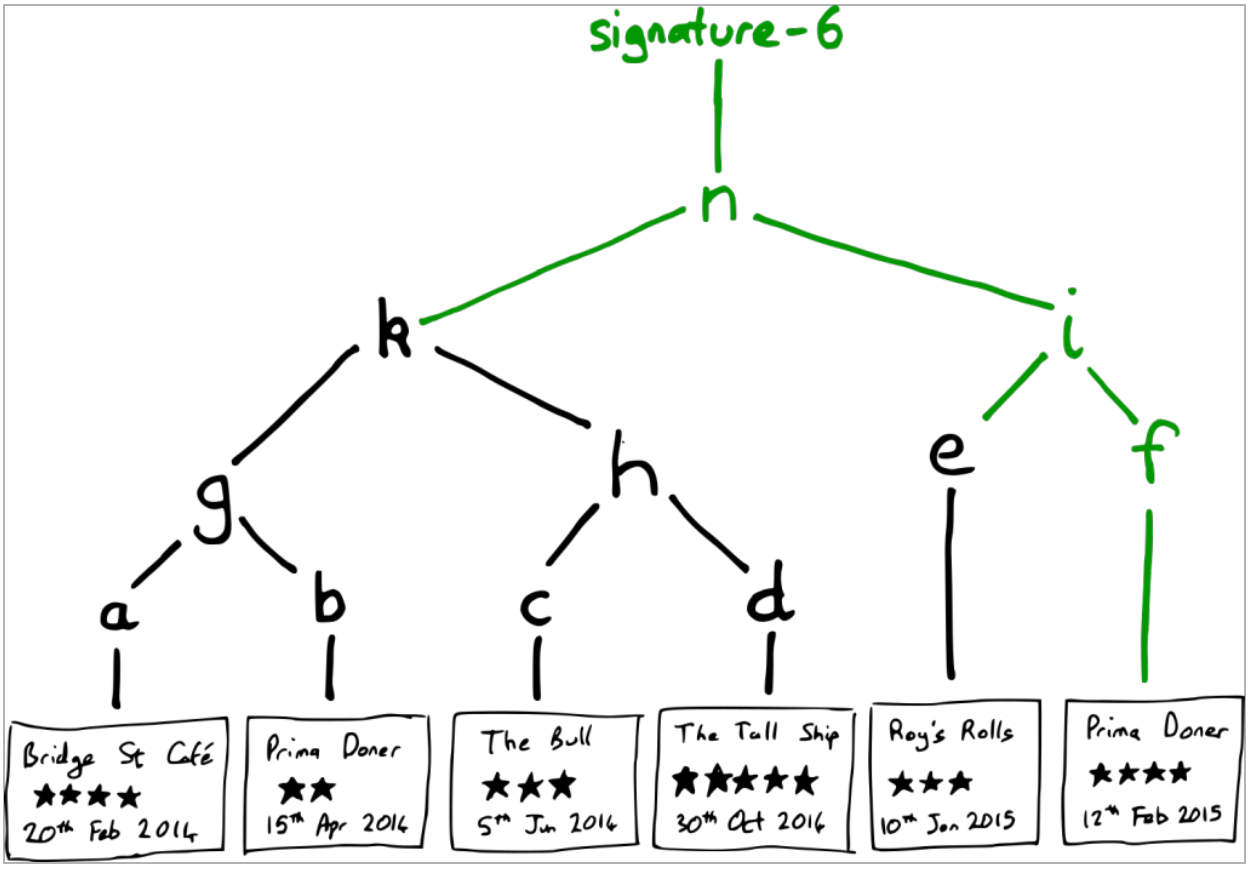
of:

- The signature (in this case, signature-5)
- The number of items in the register. From this, you can work out the tree's size and structure
- The position of the report within the tree (in this case, it is in position 5). This tells you which path to compute through the tree
- The intermediate Merkle tree hash values for siblings on the path from the report to the tree head (in this case, there is only one sibling node on the path: **k**)

Using a tree structure means that a single signature can sign the whole register, and the same signature can generate a digital proof on demand for any one entry within the register. A proof has a size in logarithmic proportion to the size of the whole register.

Adding new reports to a register

On 12th Feb 2015, Prima Doner is reinspected and found to have improved greatly, from two stars to four. We wish to add this new report to the register. To do this, we want to create a new Merkle tree which includes the new report:



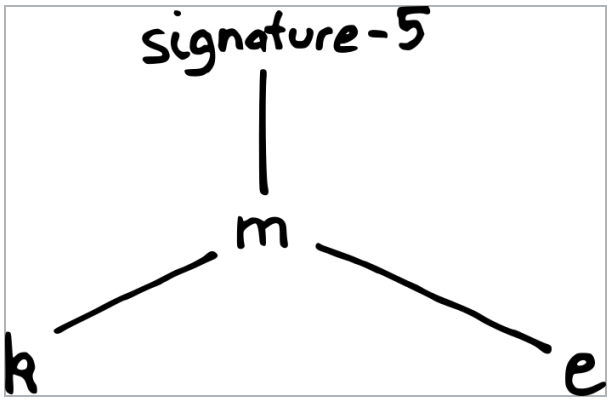
The new Merkle tree is able to reuse large parts of the previous tree. In particular,

everything from **k** and **e** downwards from the previous Merkle tree. The new parts of the Merkle tree are coloured in green.

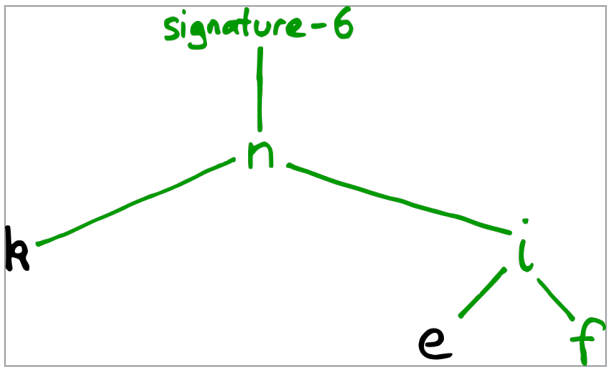
Keeping your word

As new inspection reports are recorded, it is important that we don't forget about the old reports. How can a user have confidence that a register is keeping all of its old records legitimately, and is not rewriting history? They can request a consistency proof, which demonstrates that the new Merkle tree has all of the inspection reports from the old tree, in the same order, with extra reports appended at the end.

In this example, the consistency proof is a demonstration that the first five reports in the new tree are exactly the same as all five reports from the old tree. To demonstrate this, we need the hashes from the new tree which cover the first five reports; in this case, the hashes are **k** and **e**. We can then demonstrate that **k** and **e** cover the whole of the old tree:



And that **k** and **e** cover the first five reports of the new tree:



Just like the digital proof of authenticity for a single report, we do not need to download the whole tree to demonstrate the consistency of successive versions of the register.

Conclusion

Merkle trees are a promising technology for securing the integrity of a register. They provide efficient operations for generating proofs for single records, for adding new records to a register, and for providing guarantees that data is never lost from the register.

We still have outstanding questions about integrity. There are some particularly thorny use cases which involve some sort of redaction: eg right to be forgotten, or spent criminal convictions. Balancing the needs of digital proofs with the needs of redaction is a complex subject. There will also be further needs that we haven't even discovered yet. We are just beginning to understand how to guarantee the integrity of a register.

You can follow Phil on Twitter, sign up now for email updates from this blog or subscribe to the feed.

> If this sounds like a good place to work, take a look at Working for GDS - we're usually in search of talented people to come and join the team.

Tags: registers
