de·ter
the *science* of cybersecurity

Using Attribute Based Access Control (ABAC)

**Ted Faber, Alefiya Hussain, John Wroclawski,
Mike Ryan, Ken Klingenstein,
Stephen Schwab, Jay Jacobs**

22 July 2010

USC Viterbi
School of Engineering

ISI
Information Sciences Institute

SPARTA
Pride in Performance

House Rules

- Topic: the ABAC Trust Mgt System
  - Attribute Based Trust Management System
  - Origin: Sparta (NAI at the time)
  - A Concrete Realization a General ABAC
- Workshop/Tutorial Format
  - Questions and interruptions welcome
- A Three-Hour Tour

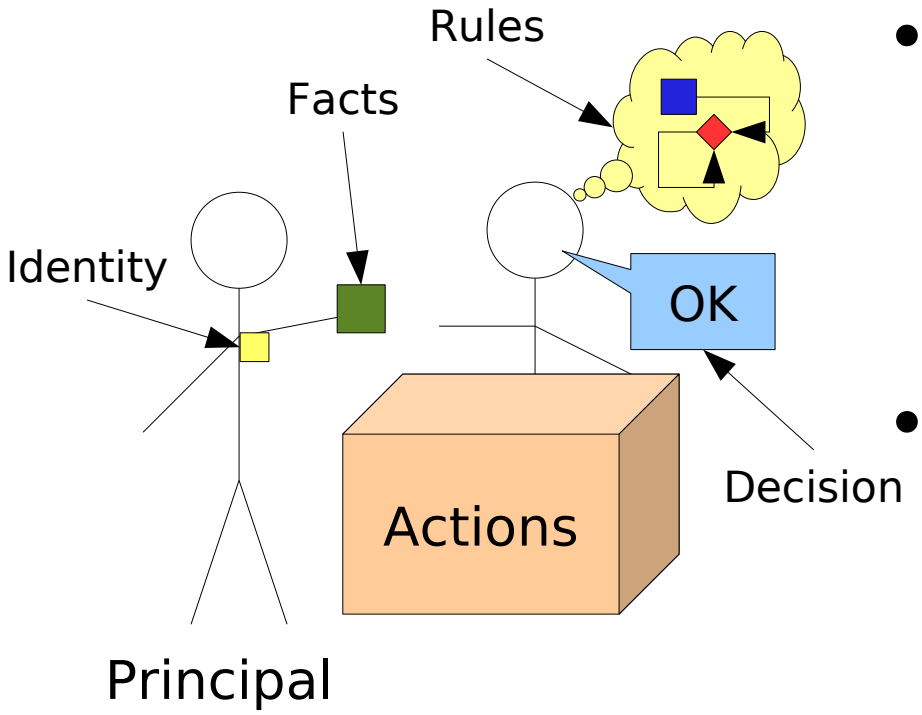
de·ter
the *science* of cybersecurity

Outline

- Authorization Problems & ABAC Features
- Using ABAC
  - Principals & Credentials
  - Policies
- Examples of ABAC in Use
- The ABAC Library and Example Code
  - Basic functionality
  - Multiple Bindings
- Future Developments

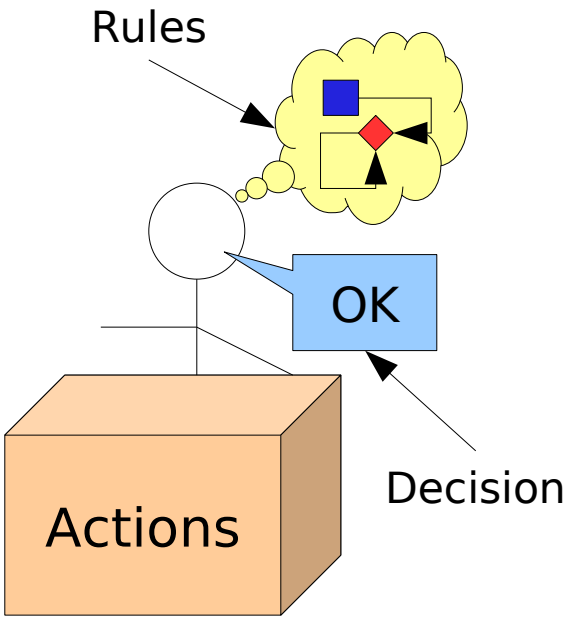
Authorization

- Basic Question: “Can Principal Perform Action?”



- Decision Quality
  - Make Right Decisions
  - Set Correct Policy
  - Audit Operation
- Scaling
  - More Users & Providers

Problems With Quality

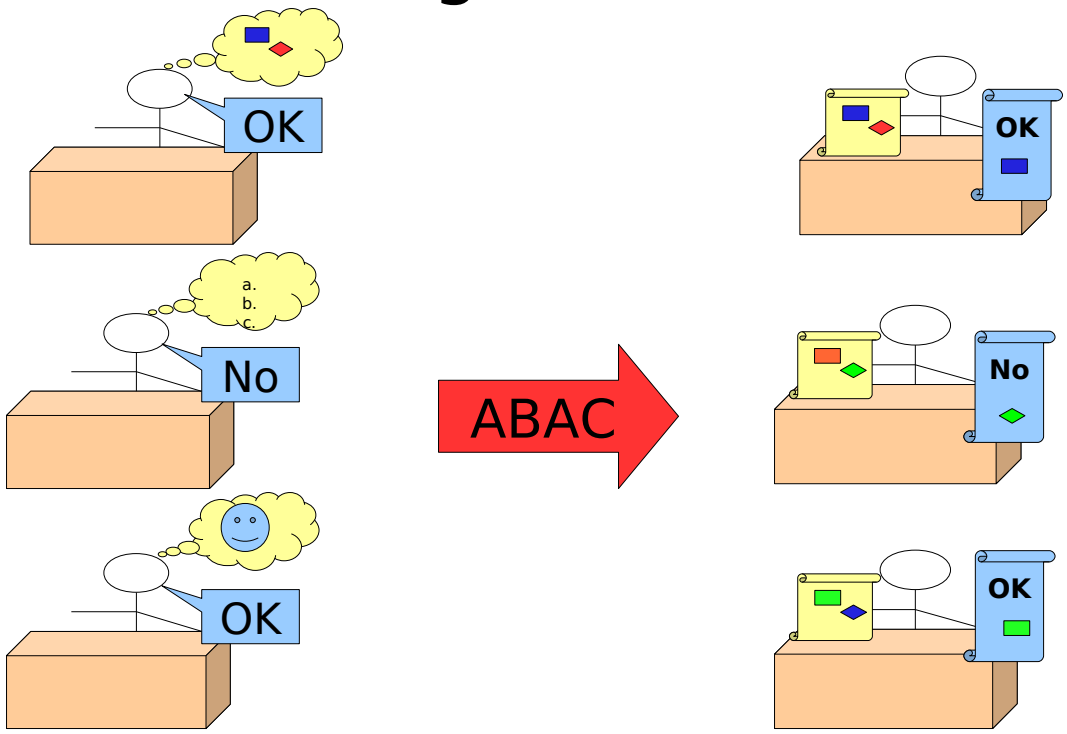


- Rules Often Part Of Application
  - Difficult To Understand
    - "Who can come in here?"
  - Difficult To Debug
    - "How did they get in here?"
- Boolean Decisions
  - Difficult To Audit
    - "Why did you let them in?"

de•ter
the *science* of cybersecurity

ABAC and Quality

- Standardize Logic For Authorization
- Make Rules Explicit
- Include Reasoning in Decisions



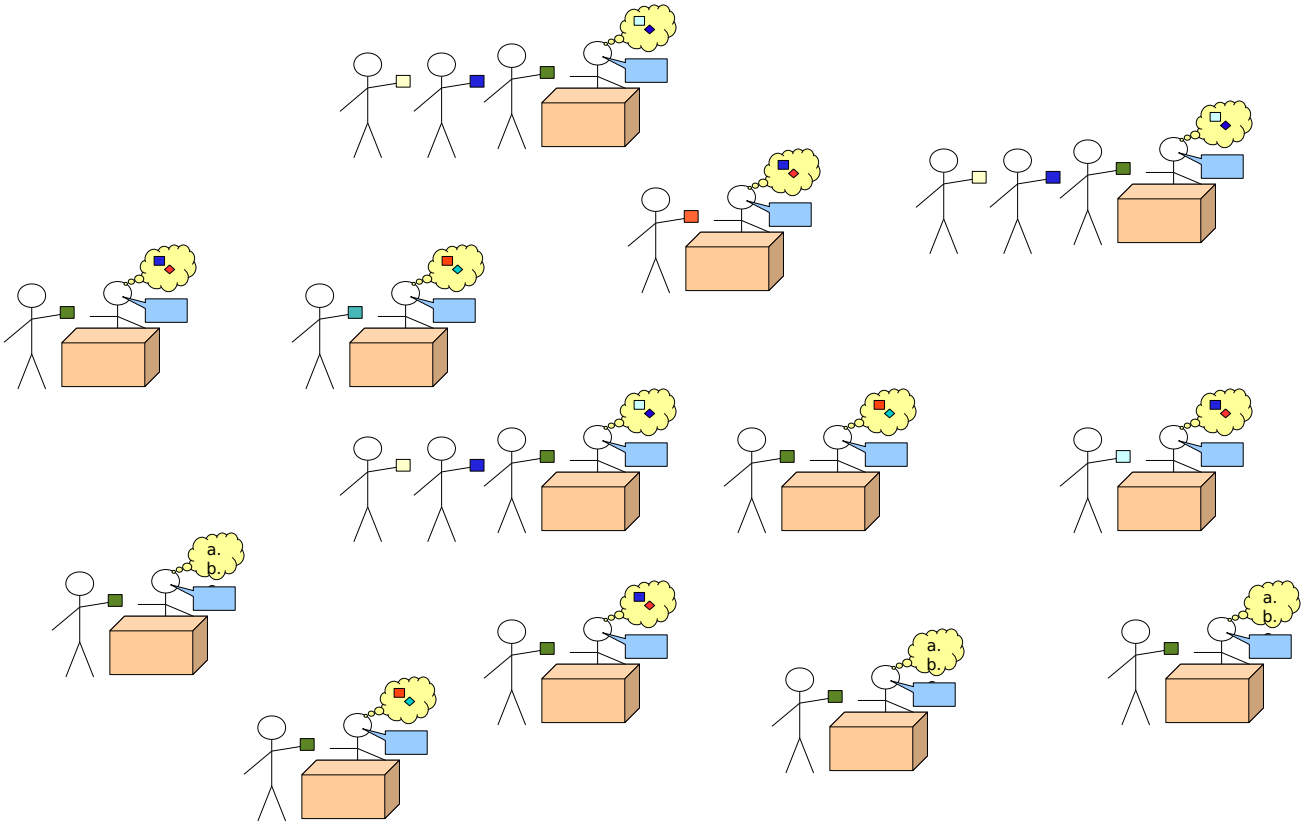
New Players: Auditors

- Policy Auditors
  - Use Explicit Rules To Check Policy
- Forensic Auditors
  - Use Explicit Reasoning To Confirm Decisions
  - Track Unexpected Authorization Decisions
- Goals:
  - Better Designed Policies
  - Better System Monitoring

Changes Auditing From Screen Scraping To Reasoning

de·ter
the *science* of cybersecurity

Problems at Scale

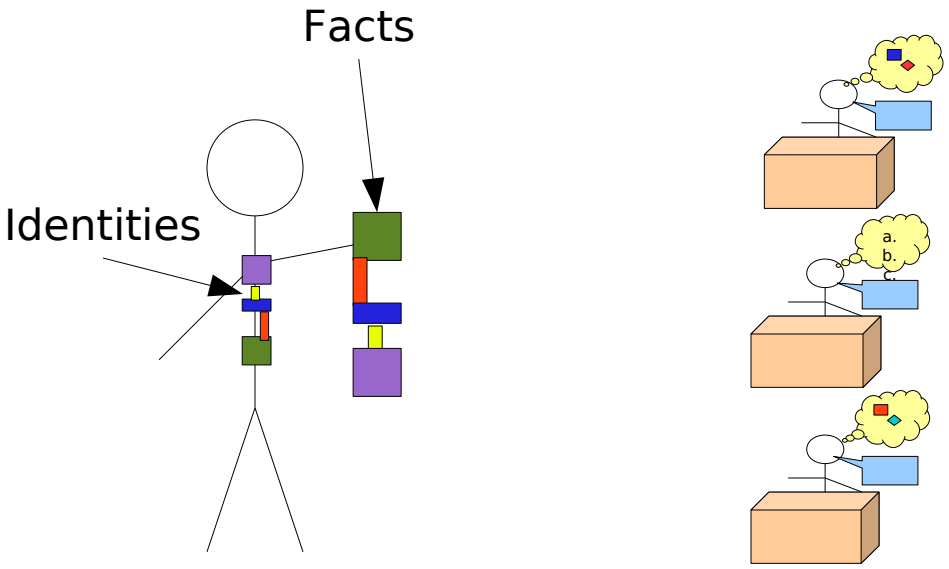


- Many incompatible authorization systems must work together

de•ter
the *science* of cybersecurity

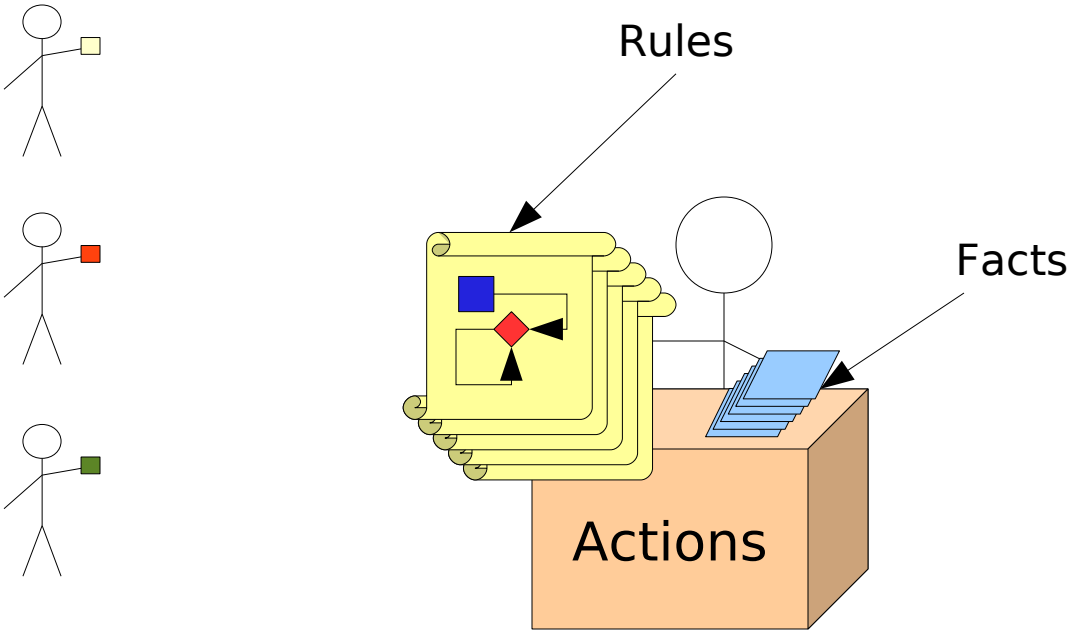
Scale: The Principal's View

- More providers/facilities means more
  - Formats For Facts
  - Ways of Proving Identity



Scale: Facility View

- More Principals means
  - Providers: More Facts/Rules To Understand
  - Admins: More Time Administering Local Info/Facts



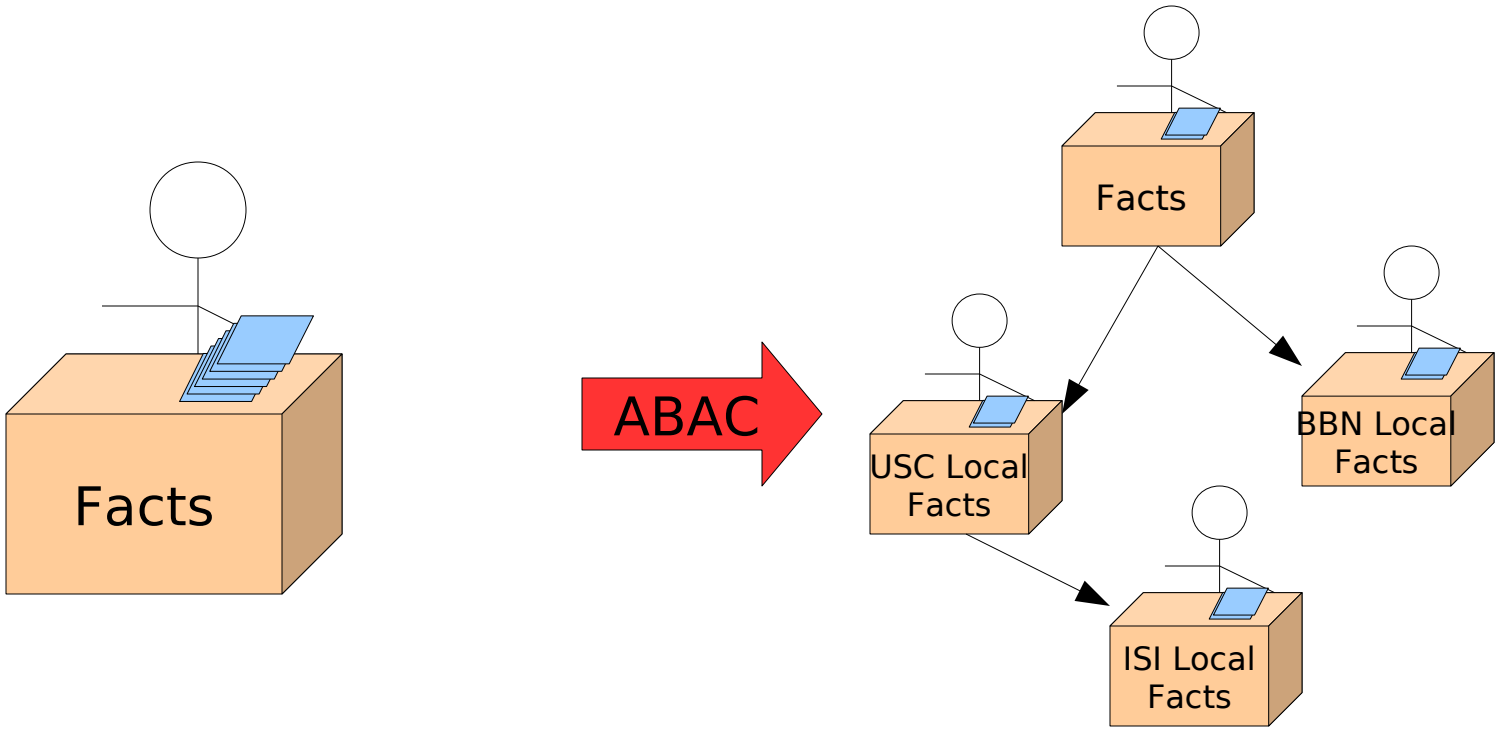
ABAC and Scaling

- Powerful Delegation
  - Decentralizes Control of Facts
  - Localizes Decisions
- New Principals
  - Certifiers Act As Fact Distributors

de•ter
the *science* of cybersecurity

Delegation Rules

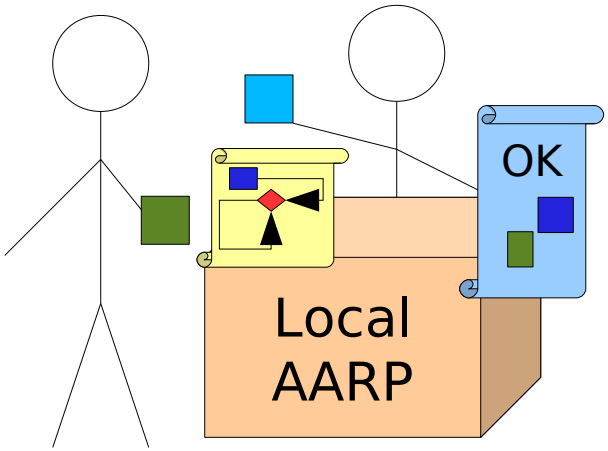
- Principals Delegate Power To Attest Facts
- Delegation Can Cross Administrative Lines
- Delegations Visible As Rules



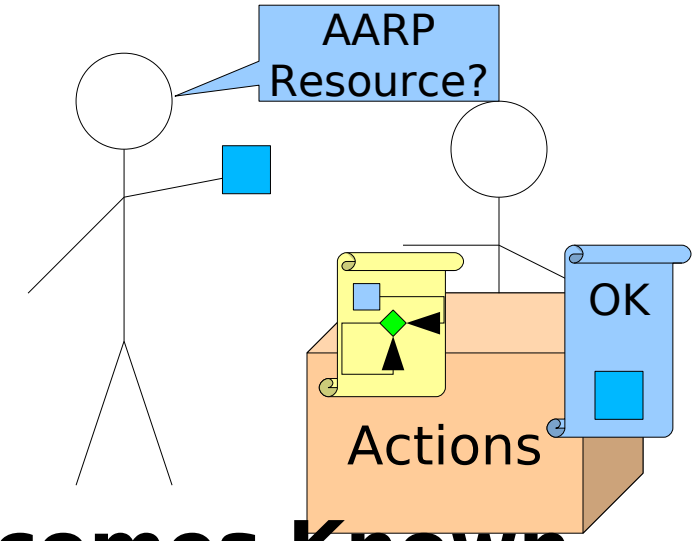
Enabling Certifiers

- Third parties that attest facts
  - Example: AARP

Local Service
Delegated Rights



Facility Respects
Certifier Fact



**Once A Certifier Becomes Known
Services Accept Credentials
Without Direct Agreement**

ABAC Certifiers and Scale

- Certifiers Help Facilities
  - Widely trusted facts to include in policy
  - Reduce local credential management
  - Anchors for federation
- Certifiers Help Principals
  - A few facts can gain access to many facilities
  - Simplify joining a coalition

Contrast With Others

- Primary Distinctions
  - Richer Delegation Rules
  - Explicit Reasoning
- Shibboleth
  - Attests Facts About Users
  - No Representation Of The Reasoning
- X.509 /SSL/PKI
  - Hierarchical Trust
  - Simple Identity & Fact Attestation
  - Again, No Reasoning

de·ter
the *science* of cybersecurity

Outline

- Authorization Problems & ABAC Features
- Using ABAC
- Examples of ABAC in Use
- The ABAC Library and Example Code
- Future Development

ABAC Model Outline

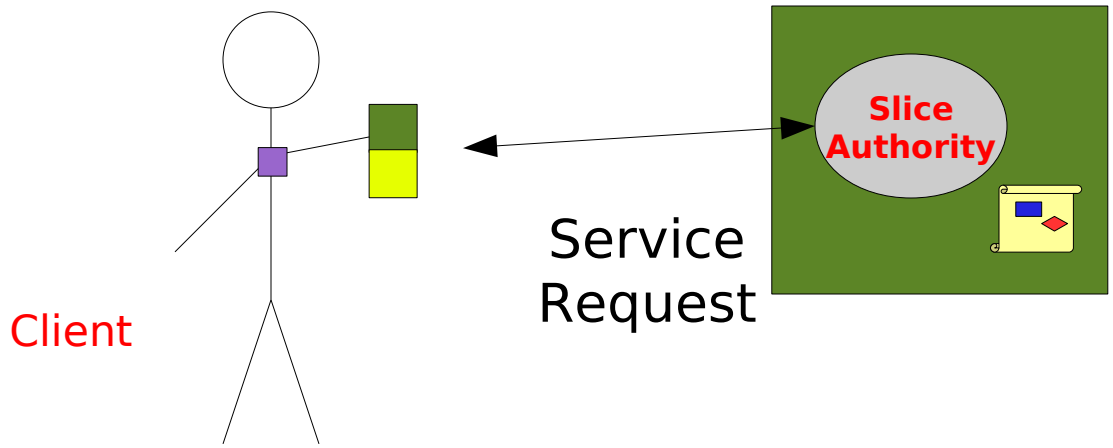
- Fundamental Objects
  - Principals, Attributes, Credentials, and Proofs
- Negotiation
  - Proving Access
- Interfacing to Applications
  - New and Legacy Applications
- ABAC Logics
  - RT0 and friends

ABAC Principals

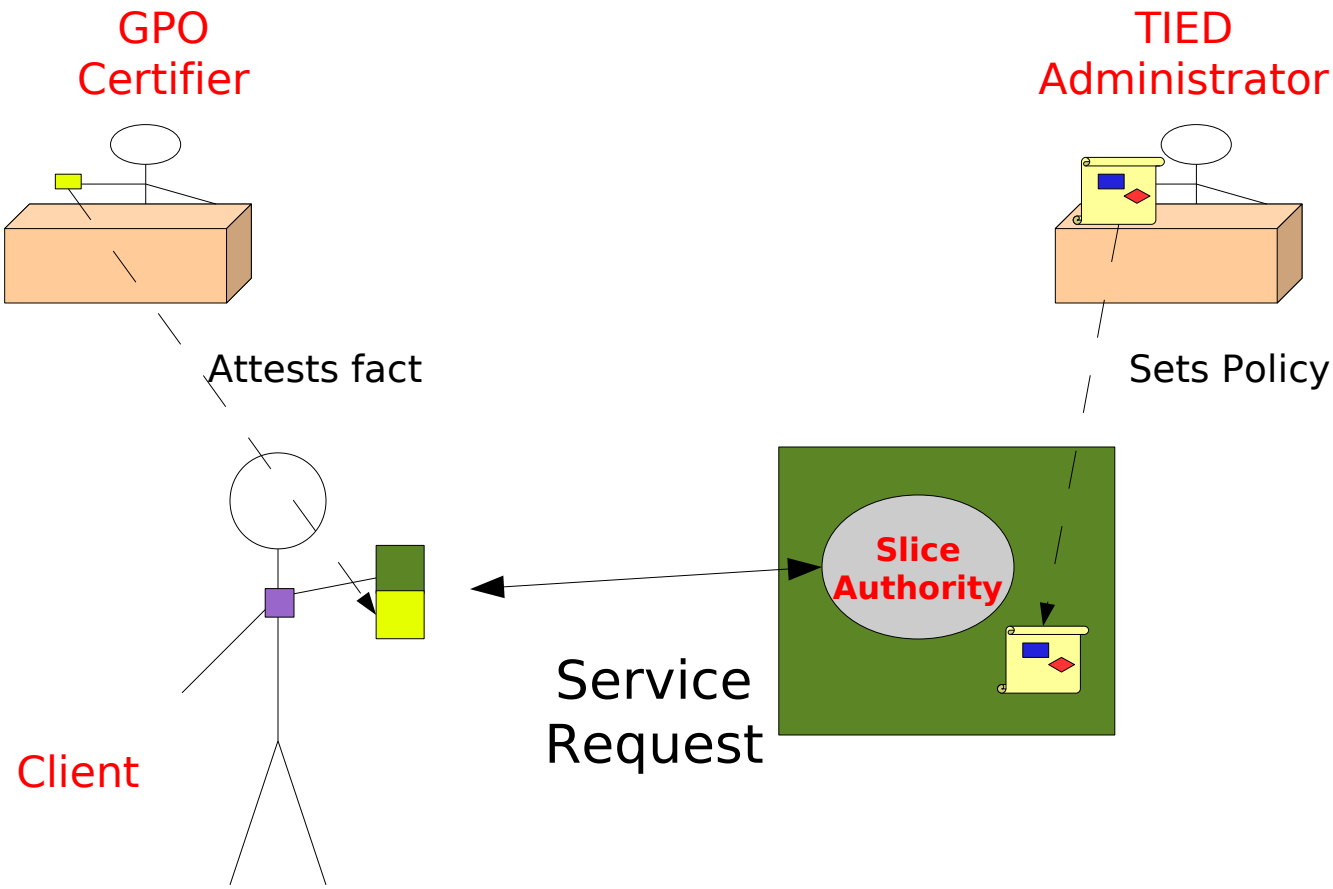
- Principals Represent
  - Requesters of Actions
    - “Can this principal take this action?”
  - Service Providers
  - Certifiers
  - Administrative Entities

de•ter
the *science* of cybersecurity

ABAC Principals In The Wild



ABAC Principals In The Wild



de•ter
the *science* of cybersecurity

ABAC Principal Identity Requirements

- A Principal Must Be Able To:
  - Prove Its Identity To Another Entity
  - Securely Attest Attributes About Principals
- A Principal's Identity Is Unique
  - If 2 Entities Refer To Principal With an ID, They Are Referring To The Same Principal
    - (One Human or Process May Be Several Principals)
- These Are The Only Constraints

Minimal Principals

- Bootstrap Identity From Many Services
- Applications Provide Semantics
  - Personal Information
  - Readable Names
- ABAC Only Knows What It Defines (Attributes)
  - No Implicit Information In Principal ID
  - Everything Available For Analysis

Attested Attributes

Principal.Role

- Role (a string) Is Attested By Principal
- Each Principal Defines An Attribute Space
  - *P.admin* differs from *Q.admin*
  - Roles Are Often Related By Convention
    - Without A Rule, This Is Immaterial
- Each Principal Controls Its Attribute Space

Attributes and Actions

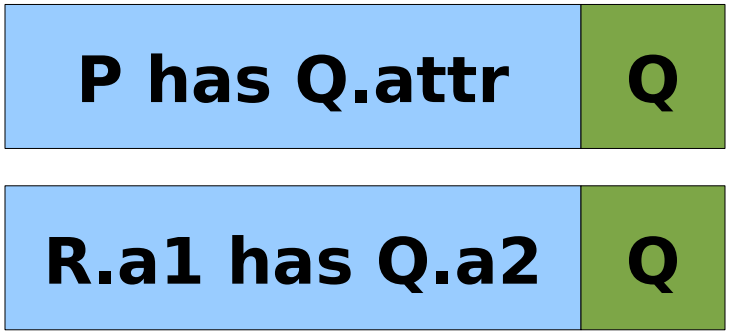
- Attributes Bound To Actions By Apps
  - "Can this principal take this action" → "Does this principal have this attribute"
- Policy Set In Terms Of Attributes
  - Attributes Attached To Principals
  - Some Control of Attributes Delegated
- Service Provider Checks Attributes
  - Binding (Attribute to Action) in configuration

Attaching Attributes

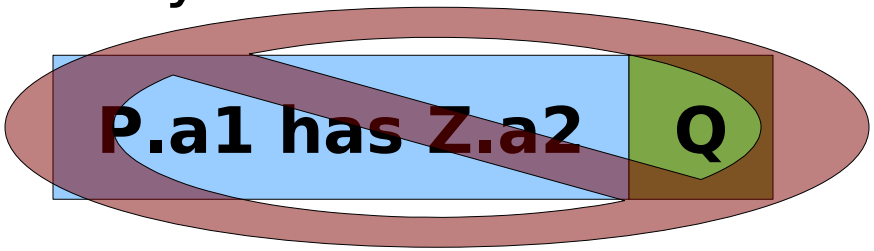
- Principal Directly Attaches
  - Q says “P has Q.attr”
    - Controlling Principal (Q) Assigns Its Attribute Space
- Principal Defines A Rule That Attaches
  - Q says “all principals with P.a1 have Q.a2”
    - Controlling Principal (Q) Delegates Some Of Its Space
- Rules Defined By ABAC Logics
  - RT0: Simple Attributes
  - RT1: Parameterized Attributes
  - …
  - RT_n is a subset of RT_{n+1}

Credentials

- Credential Manifests Assignment Or Rule



- Attested by Principal
  - Only Valid When Attested By Attribute Owner

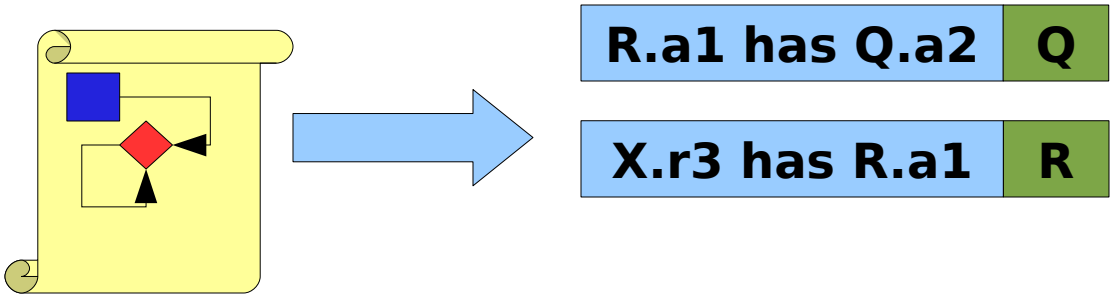


- Independently Verifiable

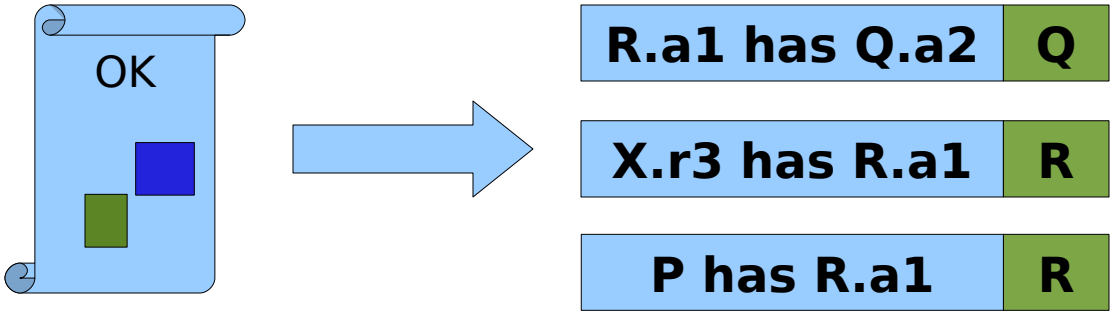
de•ter
the *science* of cybersecurity

Credential Uses

- Policies: Attribute Inference Rules



- Reasoning: Proof A Principal Has Attribute



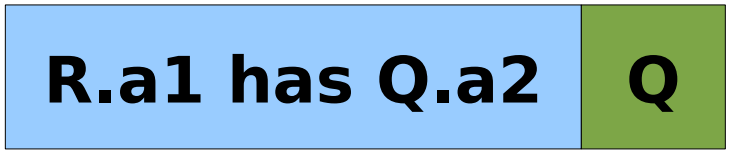
de•ter
the *science* of cybersecurity

Proofs

- Evidence That A Principal Has An Attribute
- Consists of Principal, Attribute, Credentials

P, Q.attr

P, Q.a2



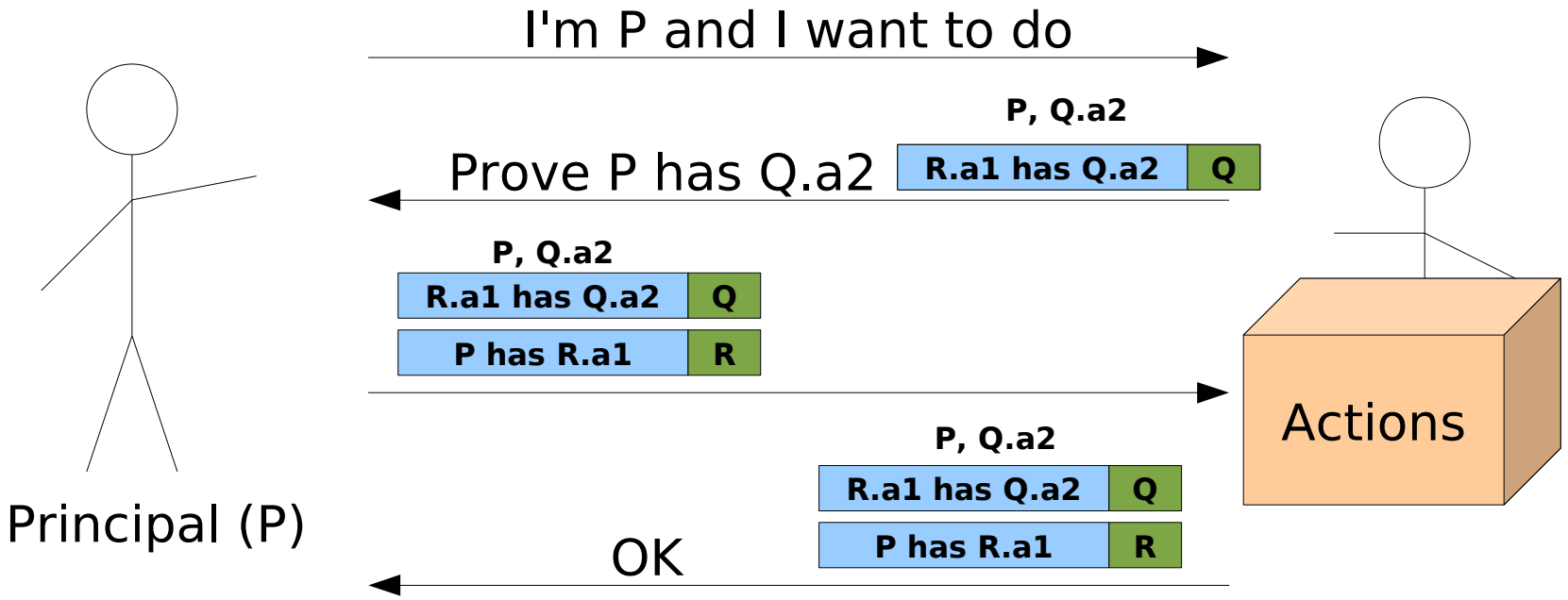
Proof Properties

- Independently Verifiable
  - Any Observer Believes The Proof
    - Auditors
    - Forensics
- Encode reasoning
  - Credentials Encode Justifications
    - Policy Checking
    - Debugging

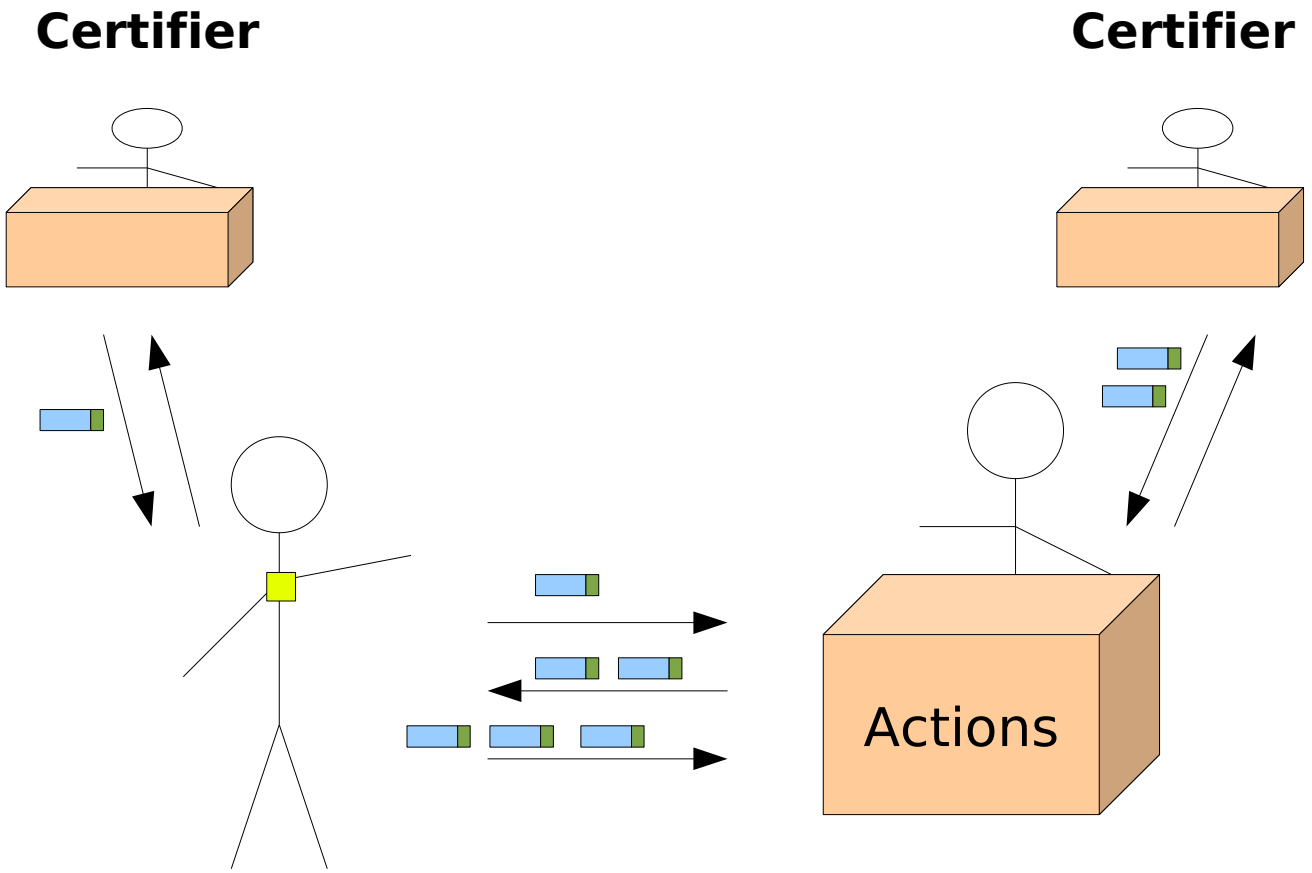
Proving

- Parties Agree On Principal And Attribute
- Exchange Credentials Until:
  - Agree On A Proof
    - Both Sides Have The Proof
  - Cannot Make Progress

Proving


I'm P and I want to do
P, Q.a2
R.a1 has Q.a2
Q
Prove P has Q.a2
P, Q.a2
R.a1 has Q.a2
Q
P has R.a1
R
Actions
P, Q.a2
R.a1 has Q.a2
Q
P has R.a1
R
OK
Principal (P)

Using Public Credentials



- Both Sides Can Gather Public Credentials
  - Credentials stand alone

Sensitive Data

- ABAC Model Supports Control Of:
  - Private Credentials
  - Sensitive Credentials
- Some Credentials Are Access Controlled
  - Show Clearance Only To Gov't Agent
- Partner Must Prove Attribute To See Credential
- Paranoia:
  - Can Ask For Proof of Attribute To Hide Missing Credential(!)

de•ter
the *science* of cybersecurity

Integrating With An Application

- ABAC Is Part Of Applications
- An Application:
  - Binds Request to Principal
    - Challenge/Response
    - Signed Request
  - Binds Service to Attribute
    - Configuration
- Carries Out An ABAC Negotiation
  - New Applications Include This Explicitly

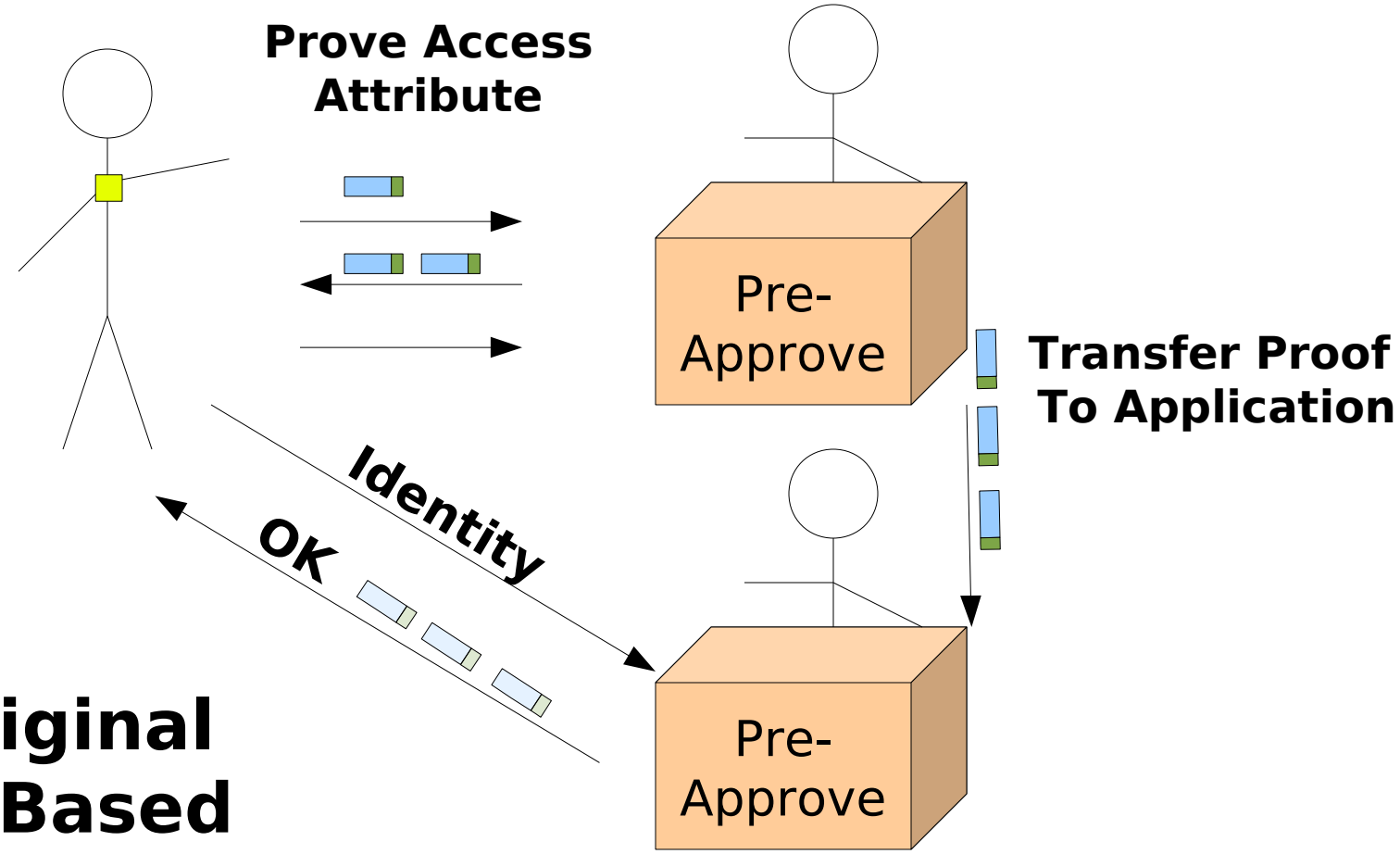
de•ter
the *science* of cybersecurity

Adding ABAC To Applications

- Existing Application With Opaque Authorization Data Fields
  - Put ABAC Credentials into Opaque Fields
  - Use Error Codes To Indicate More Info Needed
- Existing Application
  - Add Separate Pre-prover
    - Sample Code In The Library
  - Application Takes ABAC creds from Pre-prover

de•ter
the *science* of cybersecurity

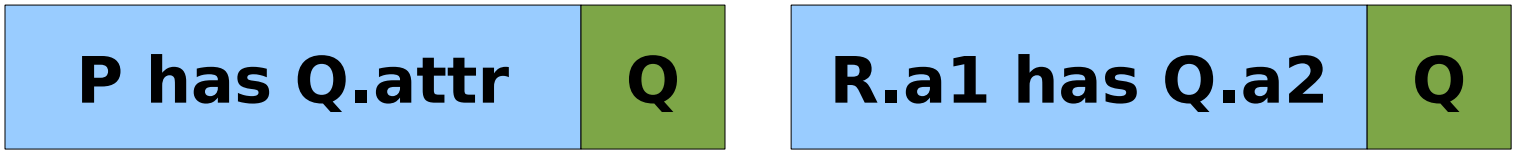
Pre-Approval



Original ID-Based Exchange

ABAC Logic: RT0

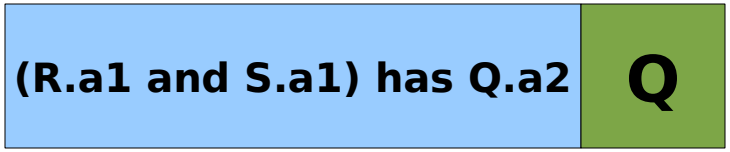
- Assignment And Delegation
  - Seen These Already:



  - Written as:
    - Q.attr ← P
    - Q.a2 ← R.a1

Intersections

- Intersection (Conjunction)

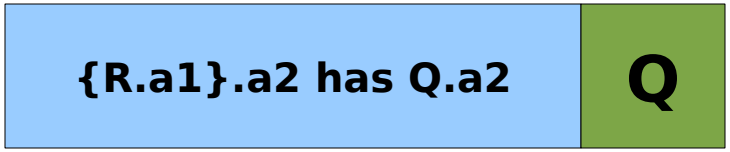


  - A Principal With R.a1 and S.a1 Also Has Q.a2
  - Not A Shorthand For 2 Credentials
  - Written:
    - Q.a2 ← R.a1 and S.a1

de•ter
the *science* of cybersecurity

RT0: Linking Credentials

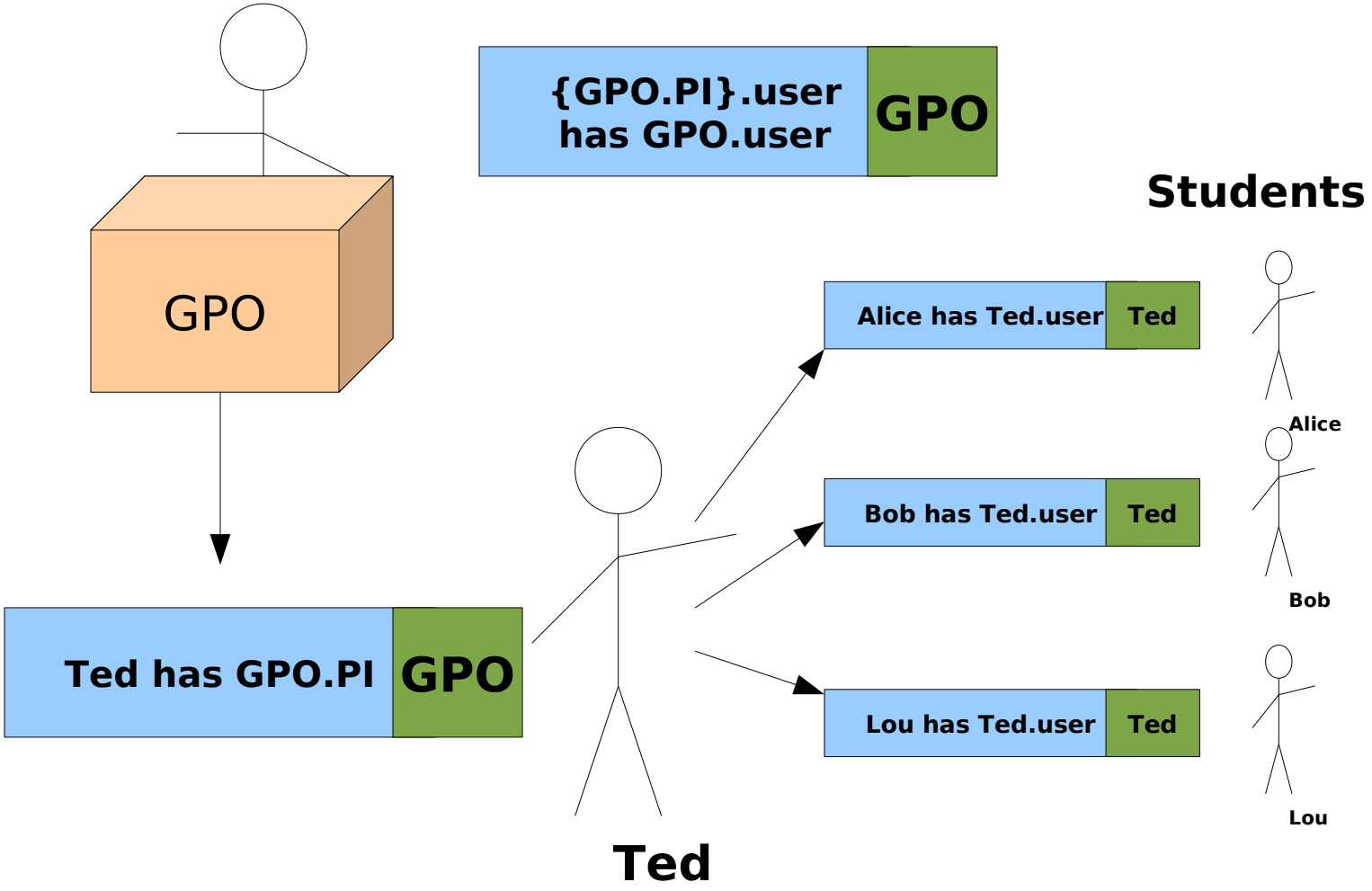
- Linking Credential
  - Delegates To A Set Of Prinicpals



  - {R.a1} Is The Set Of Principals With R.a1
    - Any Member of the Set Can Now (Indirectly) Assign Q.a2
    - Principal R Controls The Membership of {R.a1}
    - R.a1 is the Linking Role
  - Written
    - Q.a2 ← R.a1.a2
    - I Prefer: Q.a2 ← (R.a1).a2

de•ter
the *science* of cybersecurity

Linking Role Example



All The Students Have GPO.user

Linking Credentials & Principal Classes

- Principal Classes
  - Useful
  - Collaboratively Defined
    - (Some Shared Semantics Required)
- Example:
  - GPO: “All Graduate Students of GENI PIs are Blue”
  - GPO.Blue ← GPO.PI.grad_student
    GPO.PI ← Ted, GPO.PI ← Steve, …

Outline

- Authorization Problems & ABAC Features
- Using ABAC
- Examples of ABAC in Use
- The ABAC Library and Example Code
- Future Development

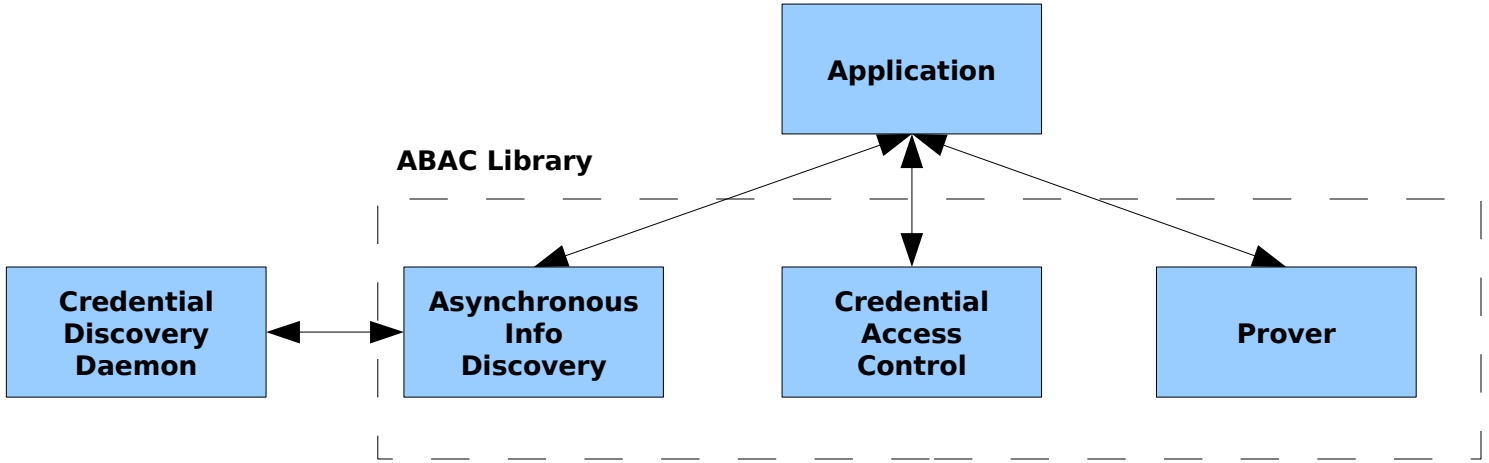
de·ter
the *science* of cybersecurity

Outline

- Authorization Problems & ABAC Features
- Using ABAC
- Examples of ABAC in Use
- The ABAC Library and Example Code
- Future Development

de·ter
the *science* of cybersecurity

Library Architecture

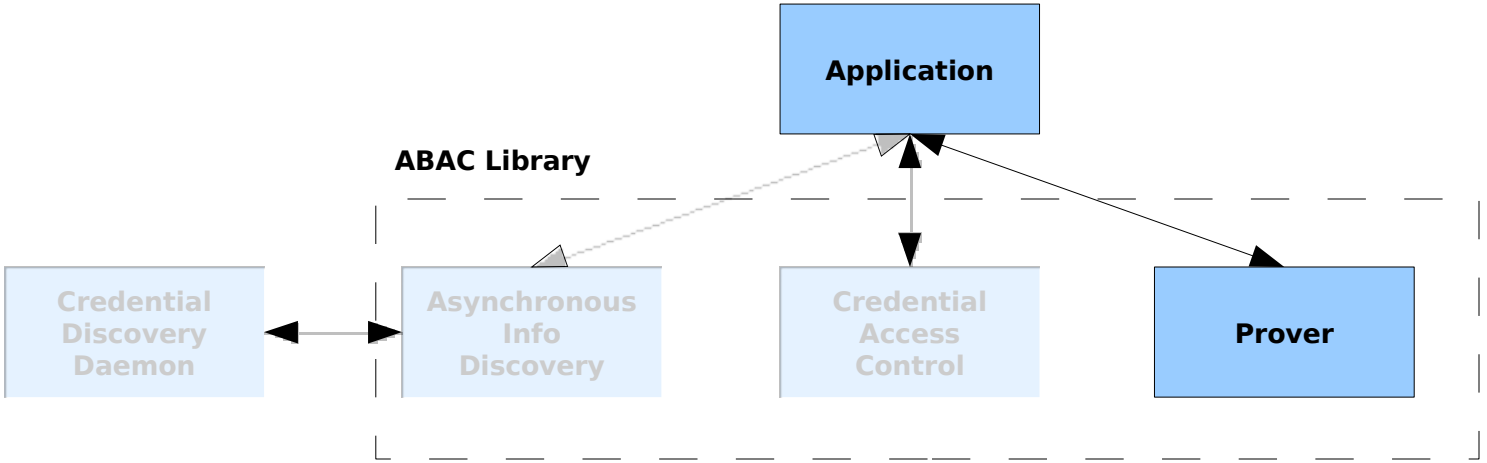


The Library

- Beta Release
  - Basic Functionality Today
    - Subsume simple ID authorization
  - Base For Expansion
- Current Features:
  - X.509-based Credential Management
  - RT0 Proofs
- Future Features
  - More Authentication Support
  - Asynchronous Credential Discovery
  - Information Protection

de·ter
the *science* of cybersecurity

Library Today



Principal Implementation

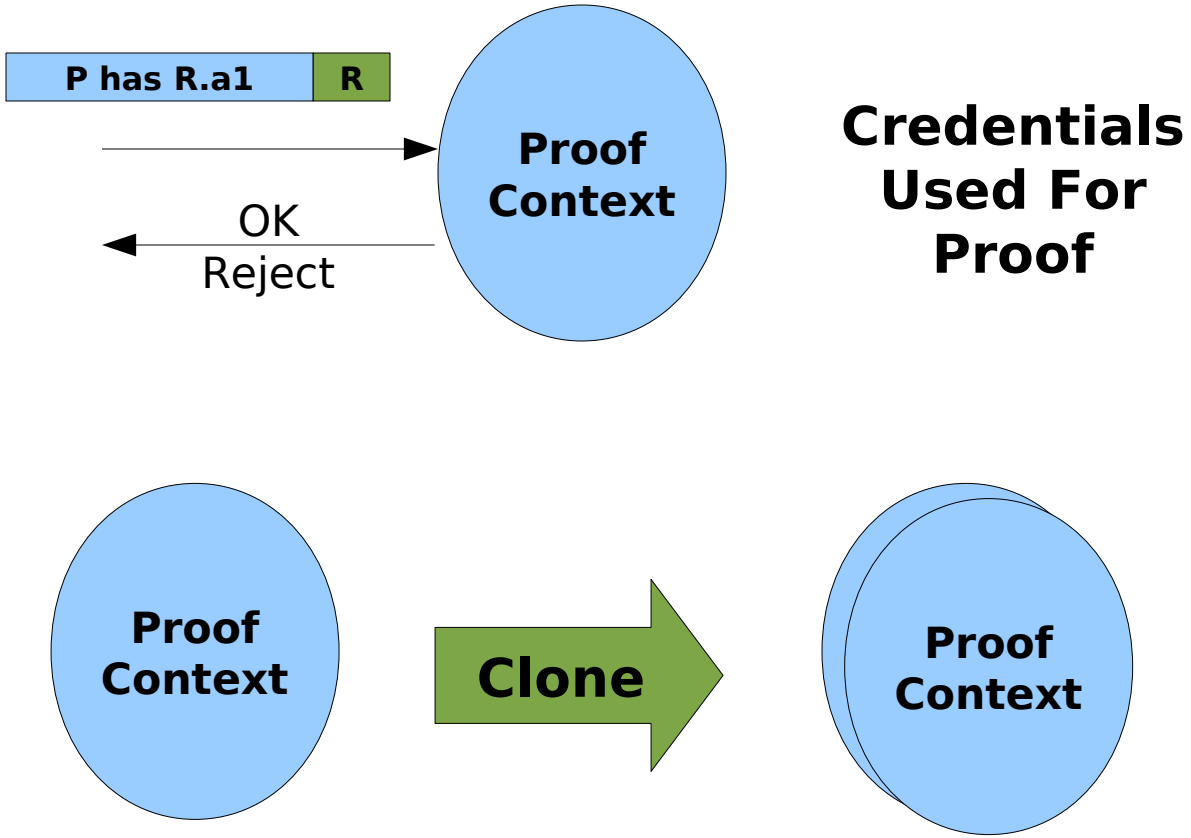
- Principal ID: Public Key Fingerprint
  - RFC 3280 fingerprint
- Currently
  - Self-signed X.509 Certificates
  - Existing SSL Libraries
    - Binds principals to requests
    - Well Tested Crypto And Challenge Base
- Future
  - Other Key-Based Authentication Is Direct
  - Non-Key-Based Requires More Infrastructure

Credential Implementation

- ABAC Cannot Reformat Credentials
  - Credential Is Signed By Prinicpal
  - Reformatting Is Forging
- Goal: Small Set Of Credential Formats
  - X.509 Attribute Certificates (RFC 3281)
  - SAML Attribute Assertions
- Today: X.509 Attribute Certificates

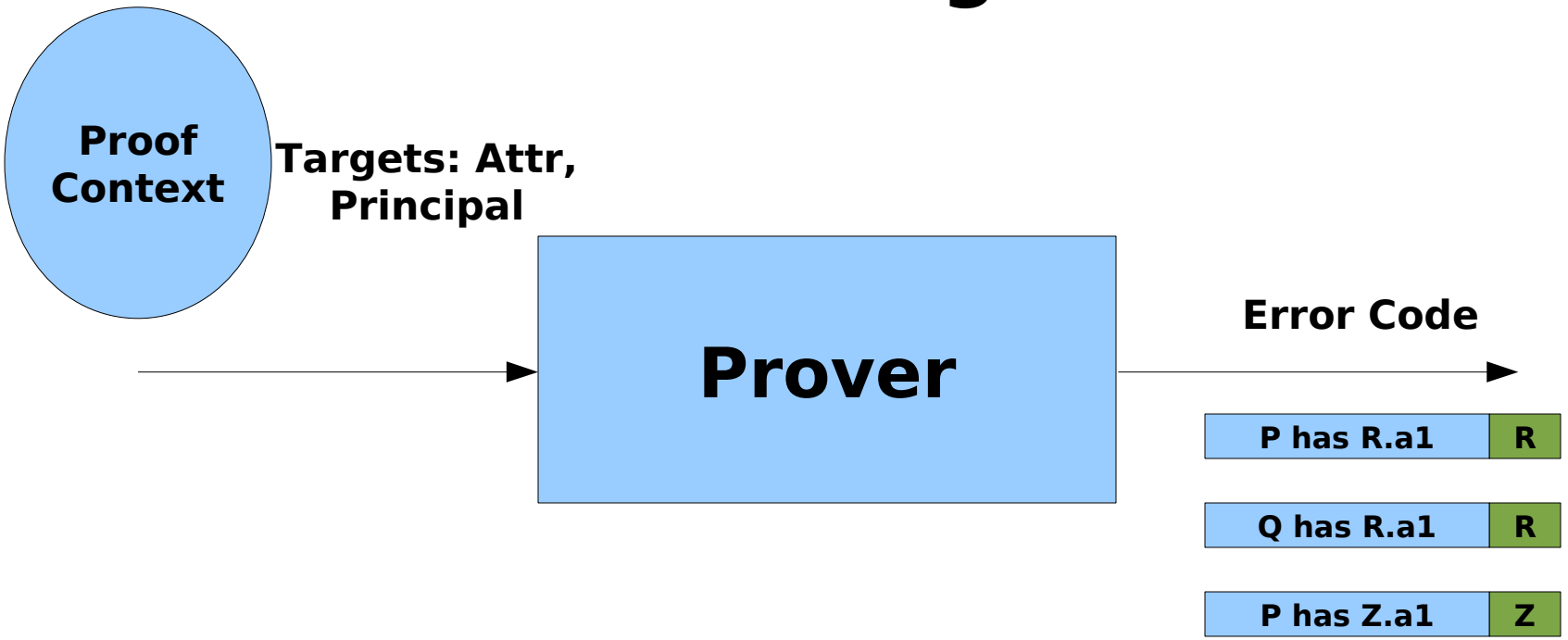
de•ter
the *science* of cybersecurity

Programming Model: Contexts



de•ter
the *science* of cybersecurity

Programming Model: Proving

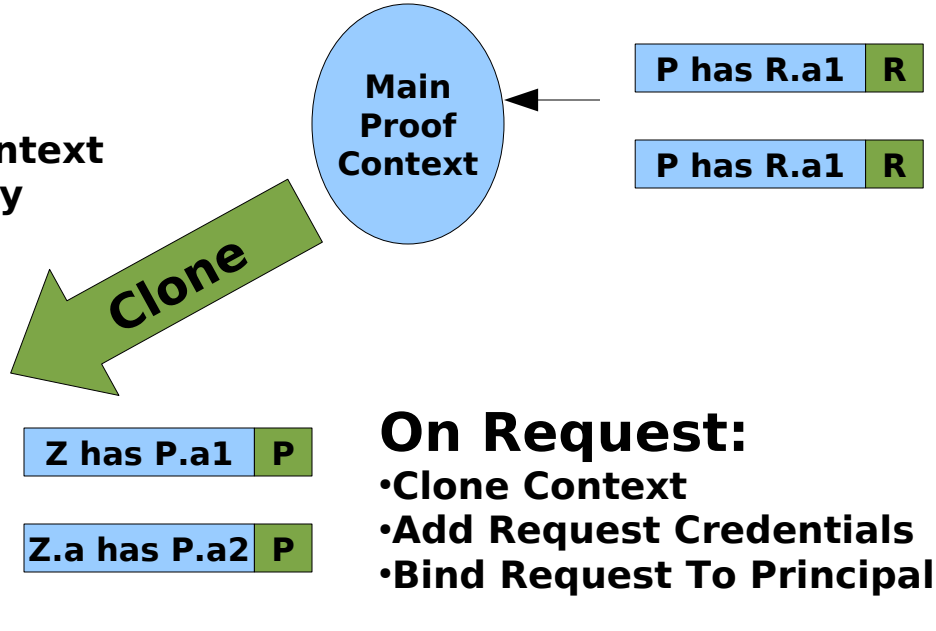


Success: Credentials Are Proof
Failure: Credentials Are Starting Point For Next Round

de•ter
the *science* of cybersecurity

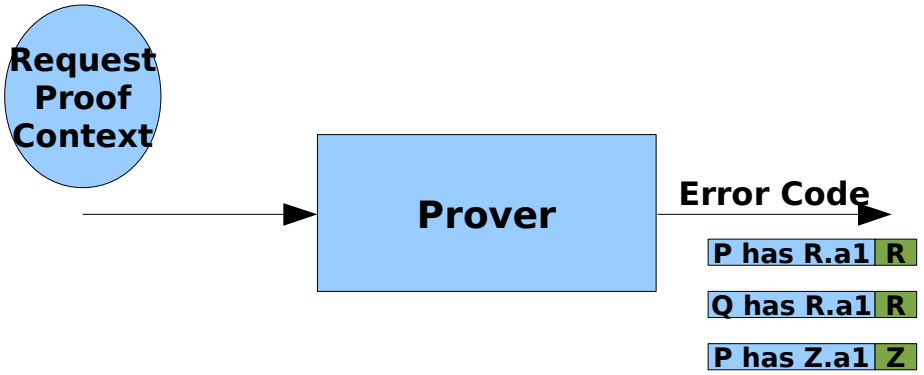
Programming Model: Application Skeleton

At Startup:
- Create Main Context
- Load With Policy



On Request:
- Clone Context
- Add Request Credentials
- Bind Request To Principal & Attr.

Process Request:
- Attempt to Prove Principal has Attr
- Return Proof or Partial
- Allow Action or Return Error
- (Release or Cache Context)



de•ter
the *science* of cybersecurity

Getting The ABAC Library

- What You Get:
  - Credential Generation Utilities
  - RT0 Prover
  - Example policies as credentials
  - Sample Pre-Prover code
  - Bindings for C, C++, Perl, Python
  - Documentation
- What You Need
  - LibStrongSwan and swig
- http://abac.deterlab.net

Pre-Installing

- Install libstrongswan (4.4.0)
  - X.509 Attribute Certificate Implementation
  - Linux IPSec
    - Only need the certificate libs, which are cross-platform
  - download/make/configure cycle
    - Just for libstrongswan
  - Details at http://strongswan.org/
- Bindings
  - Install swig 1.3
  - Standard rpm or package
  - Details at http://www.swig.org/

Installing

- Install libabac
  - download/make/configure
  - Details at http://abac.deterlab.net

Data Structures

- Context
  - Credential Validation and Proof Generation
- Credential
  - Manipulation of ABAC Attributes
- Role
  - Attribute Elements

Context: Input

- Credential Loading:
  - load_id_file(), load_id_chunk()
    - Add a principal public key
  - load_attribute_file(), load_attribute_chunk()
    - Add an attested attribute
  - load_directory()
    - Bulk load a directory of certificates
- Cloning
  - Copy Constructor (Deep Copy)

Context: Output

- Query()
  - Input: target prinicpal & attribute (strings)
  - Output: status code, Credential list
- Credentials()
  - The Contents of the Context as Credential List

Credentials & Roles

- Credential
  - Head and Tail Roles
  - attribute_cert() and id_cert() accessors
    - These access the X.509 basis for the Credentials
- Role (object before or after a ←)
  - is_role(), is_linked(), is_principal()
  - role_name(), linked_role(), principal()

Python Example

```
import sys
from ABAC import *
# Make sure arguments are present
if len(sys.argv) < 2:
    print "Usage: prover.py <keystore>"
    exit(1)
keystore = sys.argv[1]
# init library
libabac_init()
# Create Context
ctx = Context()
# Import Policies/Credentials
ctx.load_directory(keystore)

# Ask for proof
(success, credentials) = ctx.query( "3f1aca4c5911b345d81c5f1a77675dce13249d0c.fed_create",
            "5839d714b16bbe108642c5eb586c2173420bed19",)
# Print Credentials
for credential in credentials:
    print "credential %s <- %s" % (credential.head().string(), credential.tail().string())
libabac_deinit()
```

de•ter
the *science* of cybersecurity

A walk through the Pre-Prover

- Skeleton For Networked ABAC Negotiation
- XMLRPC/SSL version of Context::query()
- Code included in the libabac package

- (Also a perl ABAC example)

Server Operation

- Initialize Context
  - Read policy from credentials
- Start XMLRPC Server
- On Request
  - Get targets
    - Principal from SSL connection
    - Attribute Is Parameter
  - Clone context
  - Add new credentials
  - Prove and return

The Server

```
#!/usr/bin/perl
# Import libraries
use XMLRPC;
use ABAC;
use constant { PORT    => 8000, };

#Start ABAC
ABAC::libabac_init;

# Read Credentials into Base Context
my $keystore = shift || die "Usage: $0 <keystore>\n";
my $ctx = ABAC::Context->new;
$ctx->load_directory($keystore);

# XMLRPC startup
my $server = XMLRPC->new();
$server->add_method({
    name        => 'abac.query',
    code        => \&abac_query,
    signature   => [ 'struct struct' ],
});
$server->run(8000);
```

de·ter
the *science* of cybersecurity

Server Responder

```
sub abac_query {
    my ($server, $request) = @_;

    my $peer_cert = $server->{peer_cert};
    my $peer_id = ABAC::SSL_keyid($peer_cert);

    # clone the context so the state remains pure between requests
    my $local_ctx = ABAC::Context->new($ctx);
    foreach my $cred (@{$request->{credentials}}) {
        # Import request credentials into the clone
        my $ret = $local_ctx->load_id_chunk($cred->{issuer_cert});
        warn "Invalid issuer certificate" unless $ret == $ABAC::ABAC_CERT_SUCCESS;

        $ret = $local_ctx->load_attribute_chunk($cred->{attribute_cert});
        warn "Invalid attribute certificate" unless $ret == $ABAC::ABAC_CERT_SUCCESS;
    }

    my $role = $request->{role};
    # Do the proof and return the results
    my ($success, $credentials) = $local_ctx->query($role, $peer_id);

    return $success;
}
```

Credential Manipulation

- Creddy is Credential Manipulation Tool
- Functions:
  - Create – make a new identity
  - Attribute – make a new credential
  - Verify – confirm validity of credential
  - Keyid – get the fingerprint/ID
  - Roles – list the roles in an attribute credential

Creddy: Create

- Making A New ID

  `$ creddy --generate --cn=`*`name`*
- Results
  - New X.509 cert in *name*`_ID.pem`
    - Self-signed
  - New private key in *name*`_private.pem`

```
$ ./creddy/creddy --generate --cn=test
Generating key, this will take a while. Create entropy!
    - move the mouse
    - generate disk activity (run find)
$ ls test*
test_ID.pem      test_private.pem
```

- Congratulations: You're A Principal

Creddy: Verify

- Access to same validation as Context
- Example:

```
$ creddy/creddy --verify --cert=test_ID.pem
signature good, certificates valid
```

- With a bad certificate:

```
$ creddy/creddy --verify --cert=test2_ID.pem
signature invalid
```

Creddy: Get Principal ID

- Principal IDs used in credentials
- Running the command:

```
$ creddy --keyid --cert=test_ID.pem
c4c1a11fc17e10efb5951866cd073052fde3a764
```

de·ter
the *science* of cybersecurity

Creddy: Assign an Attribute

Assign create to another principal from our new principal's attribute space

```
$ creddy --keyid --cert=test_ID.pem
c4c1a11fc17e10efb5951866cd073052fde3a764
$ creddy --keyid --cert=subject_ID.pem
Bcecc370fa6b01cdca4a8876bd3ca93d494b9877
$ creddy --attribute --issuer=test_ID.pem --key=test_private.pem
--subject=subject_ID.pem --role=create --out=assign.der
$ creddy --roles --cert=assign.der
c4c1a11fc17e10efb5951866cd073052fde3a764.create <-
bcecc370fa6b01cdca4a8876bd3ca93d494b9877
```

de•ter
the *science* of cybersecurity

Creddy: Delegate an Attribute

Delegate create to another principal's researcher attribute from our new principal's attribute space

```
$ creddy --attribute --issuer=test_ID.pem --key=test_private.pem
--subject=subject_ID.pem --subject-role=researcher --role=create
--out=delegate.der

$ creddy --roles --cert=delegate.der

c4c1a11fc17e10efb5951866cd073052fde3a764.create <-
bcecc370fa6b01cdca4a8876bd3ca93d494b9877.researcher
```

de•ter
the *science* of cybersecurity

Creddy: Delegate to a Linking Role

Delegate **create** to a set of principals assigned the **researcher** role from a principal with another principal's **funder** role from our new principal's attribute space

```
$ creddy --attribute --issuer=test_ID.pem --key=test_private.pem
--subject=subject_ID.pem --subject-role=funder.researcher --role=create
--out=linked.der

$ creddy --roles --cert=linked.der

c4c1a11fc17e10efb5951866cd073052fde3a764.create <-
bcecc370fa6b01cdca4a8876bd3ca93d494b9877.funder.researcher
```

de·ter
the *science* of cybersecurity

Outline

- Authorization Problems & ABAC Features
- Using ABAC
- Examples of ABAC in Use
- The ABAC Library and Example Code
- Future Development

The Future

- Continuing ABAC library Development
  - SAML/Shib attributes
  - More complex logics
  - More bindings
  - More utilities
- Bigger Stuff

de•ter
the *science* of cybersecurity

What's Missing: Attribute Infrastructure

- Certifiers
  - Well-known certifiers of users
  - Well-known certifiers of facilities
  - Policies for scalable attribute assignment
- Attributes
  - Few, well-understood attributes
  - Anchors for new facilities/users

Policy Tools

- Policy Visualization and Configuration
  - Who can do what
  - Why can they
- Logging Visualization and Auditing
- Certificates are exchange format
- Users Prefer Better Abstractions

Wrapup

- Described and Motivated the ABAC model
  - Powerful formal logic
  - Attribute based semantics
  - Rich delegation power
- Showed ABAC library
  - Real code
  - Ongoing development
- Future Needs
  - Policy tools
  - Infrastructure