golang支付宝极简代码版
==========

废话不多说，直接上代码。

各种处理证书的函数

```
// 从证书里提取公钥
// 这里需要的证书是alipayCertPublicKey_RSA2.crt
// 这个函数验签的时候要用
func getPublickKey(ctx context.Context) string {
    block, _ := pem.Decode([]byte("**复制粘贴过来**"))
    if block == nil {
        return ""
    }
    cert, err := x509.ParseCertificate(block.Bytes)
    if err != nil {
        return ""
    }
    pubKeyBytes, err :=
x509.MarshalPKIXPublicKey(cert.PublicKey.(*rsa.PublicKey))
    if err != nil {
        return ""
    }
    pem := pem.EncodeToMemory(&pem.Block{Type: "PUBLIC KEY",
Bytes: pubKeyBytes})
    return string(pem)
}
```

```
// 提取应用证书序列号
// 这里需要的证书是appCertPublicKey_20210041396xxxx.crt
func getAppCertSN(ctx context.Context) string {
    block, _ := pem.Decode([]byte("**复制粘贴过来**"))
    if block == nil {
        return ""
    }
    cert, err := x509.ParseCertificate(block.Bytes)
```

```
    if err != nil {
        return ""
    }
    return md5Hash(fmt.Sprintf("%s%s", cert.Issuer, cert.SerialNumber))
}

```

```
// 提取根证书序列号
// 这里需要的证书是alipayRootCert.crt
// crt文件内容处理一下，按-----BEGIN CERTIFICATE----- -----END
CERTIFICATE-----变成[]string
func getCertRootSN(ctx context.Context) string {
    var sn bytes.Buffer
    for _, alipay_root_cert := range []string{"**复制粘贴过来**"} {
        if block, _ := pem.Decode([]byte(alipay_root_cert)); block != nil {
            cert, err := x509.ParseCertificate(block.Bytes)
            if err != nil {
                continue
            }
            // 只取SHA256WithRSA的证书
            if cert.SignatureAlgorithm != x509.SHA256WithRSA {
                continue
            }
            si := fmt.Sprintf("%s%s", cert.Issuer, cert.SerialNumber)
            if sn.String() == "" {
                sn.WriteString(md5Hash(si))
            } else {
                sn.WriteString("_")
                sn.WriteString(md5Hash(si))
            }
        }
    }
    return sn.String()
}

```

处理完证书，剩下的就是签名和验签

```
// alipay.trade.app.pay(app支付接口2.0)
//
https://opendocs.alipay.com/open/cd12c885_alipay.trade.app.pay?path
```

```
Hash=ab686e33&scene=20&ref=api
// 返回的orderStr给客户端
func AlipayAppPay(ctx context.Context) (string, error) {
    bizParams := map[string]interface{}{
        "out_trade_no": "1",
        "total_amount": 0.01,
        "subject":      "SUBJECT",
        "time_expire":  time.Now().Add(time.Minute *
15).Format(time.DateTime),
    }
    bizBytes, _ := json.Marshal(bizParams)
    bizString := string(bizBytes)

    commonParams := map[string]interface{}{
        "alipay_root_cert_sn": getCertRootSN(ctx),
        "app_cert_sn":         getCertSN(ctx),
        "app_id":              "APPID",
        "method":              "alipay.trade.app.pay",
        "format":              "JSON",
        "charset":             "utf-8",
        "sign_type":           "RSA2",
        "timestamp":           time.Now().Format(time.DateTime),
        "version":             "1.0",
        "notify_url":          "NOTIFY_URL",
        "biz_content":         bizString,
    }

    // 排序并拼接
    query := buildQuery(commonParams)

    // 签名
    sign := signature(ctx, []byte(query))

    values := buildValues(commonParams)
    values.Set("sign", sign)

    return values.Encode(), nil
}

```


```
// alipay.trade.wap.pay(手机网站支付接口2.0)
//
https://opendocs.alipay.com/open/29ae8cb6_alipay.trade.wap.pay?path
Hash=1ef587fd&ref=api&scene=21
// 拿到的formStr给h5
```

```
func AlipayWapPay(ctx context.Context) (string, error) {
    bizParams := map[string]interface{}{
        "out_trade_no": "1",
        "total_amount": 0.01,
        "subject":      "SUBJECT",
        "product_code": "QUICK_WAP_WAY",
        "time_expire":  time.Now().Add(time.Minute *
15).Format(time.DateTime),
        "quit_url":     "QUIT_URL",
    }
    bizBytes, _ := json.Marshal(bizParams)
    bizString := string(bizBytes)

    commonParams := map[string]interface{}{
        "alipay_root_cert_sn": getCertRootSN(ctx),
        "app_cert_sn":         getCertSN(ctx),
        "app_id":              "APPID",
        "method":              "alipay.trade.wap.pay",
        "format":              "JSON",
        "charset":             "UTF8",
        "sign_type":           "RSA2",
        "timestamp":           time.Now().Format(time.DateTime),
        "version":             "1.0",
        "notify_url":          "NOTIFY_URL",
        "biz_content":         bizString,
        "return_url":          "RETURN_URL",
    }

    // 排序并拼接
    query := buildQuery(commonParams)

    // 签名
    sign := signature(ctx, []byte(query))

    delete(commonParams, "biz_content")

    values := buildValues(commonParams)
    values.Set("sign", sign)

    url := fmt.Sprintf("%s?%s", config.C.Alipay.OpenApi, values.Encode())
    bizString = template.HTMLEscapeString(bizString)

    html := `<form id="alipaysubmit" name="alipaysubmit" action="%s"
method="POST"><input type="hidden" name="biz_content"
value="%s"/><input type="submit" value="ok"
style="display:none;"></form><script>document.forms["alipaysubmit"].s
ubmit();</script>`
    return fmt.Sprintf(html, url, bizString), nil
```

}

```

```
// 异步验签
// notify是阿里post请求body
// gin框架可以这样取bodyBytes, err := io.ReadAll(c.Request.Body)
func AlipayVerifyNotify(ctx context.Context, notify string) bool {
    values, _ := url.ParseQuery(notify)
    var sign string
    m := map[string]interface{}{}
    for k, v := range values {
        if k == ”sign” {
            sign = v[0]
            continue
        }
        if k == ”sign_type” {
            continue
        }
        m[k] = v[0]
    }
    query := buildQuery(m)
    return verifySign(ctx, query, sign)
}
```

```

```
// 公钥验签
// 这里的公钥是alipayCertPublicKey_RSA2.crt里提取的公钥
func verifySign(ctx context.Context, data, sign string) bool {
    publicKey := getPublickKey(ctx)
    block, _ := pem.Decode([]byte(publicKey))
    if block == nil {
        return false
    }

    pubKey, err := x509.ParsePKIXPublicKey(block.Bytes)
    if err != nil {
        return false
    }

    hash := sha256.Sum256([]byte(data))
    signBytes, err := base64.StdEncoding.DecodeString(sign)
    if err != nil {
```

```
        return false
    }

    err = rsa.VerifyPKCS1v15(pubKey.(*rsa.PublicKey), crypto.SHA256,
hash[:], signBytes)
    if err != nil {
        return false
    }
    return true
}

```

```
// 私钥签名
// 这里的私钥是应用私钥RSA2048-敏感数据，请妥善保管.txt
// txt文件里不是标准私钥格式，需要自行处理一下
// 每64位换行，开头结尾补-----BEGIN PRIVATE KEY----- -----END
PRIVATE KEY-----
func signature(ctx context.Context, data []byte) string {
    block, desc := pem.Decode([]byte(config.C.Alipay.PrivateKey))
    if block == nil {
        return ""
    }

    priKey, parseErr := x509.ParsePKCS8PrivateKey(block.Bytes)
    if parseErr != nil {
        return ""
    }

    hash := sha256.Sum256([]byte(data))
    sign, err2 := rsa.SignPKCS1v15(rand.Reader, priKey.(*rsa.PrivateKey),
crypto.SHA256, hash[:])
    if err2 != nil {
        return ""
    }

    return base64.StdEncoding.EncodeToString(sign)
}

```

辅助函数

```
```

```
// 按字母升序排序，拼接成 key=value&key=value
func buildQuery(data map[string]interface{}) string {
    keys := make([]string, 0, len(data))
    for k := range data {
        keys = append(keys, k)
    }

    // 对键进行排序
    sort.Strings(keys)

    // 按照键的顺序构造排序后的字符串
    var sortedData strings.Builder
    for i, k := range keys {
        if i > 0 {
            sortedData.WriteString("&")
        }
        sortedData.WriteString(fmt.Sprintf("%s=%v", k, data[k]))
    }
    return sortedData.String()
}

// map[string]interface{}转url.Values
func buildValues(data map[string]interface{}) url.Values {
    values := url.Values{}
    for k, v := range data {
            values.Set(k, fmt.Sprint(v))
    }
    return values
}

// 简单的md5
func md5Hash(text string) string {
    hasher := md5.New()
    hasher.Write([]byte(text))
    md5 := hex.EncodeToString(hasher.Sum(nil))
    return md5
}

```

有问题留言。

原文链接: https://juejin.cn/post/7360878754283716634